Universitat Autònoma
de Barcelona

Desarrollo del nuevo

karat Starter Kit

Memoria del proyecto de
Ingeniería Técnica en
Informática de Sistemas
realizado por

Carlos Ming Martínez

y dirigido por

Jordi Pons Aróztegui

Escola d'Enginyeria

Sabadell, julio de 2011

Jordi Pons Aróztegui,
profesor de la Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección por **Carlos Ming Martínez**

Y para que conste firma la presente.
Sabadell, **julio** de **2011**

Firmado: **Jordi Pons Aróztegui**

Ezequiel Parra Mestre, de la empresa UNIT4,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección por **Carlos Ming Martínez**

Y para que conste firma la presente.
Sabadell, **julio** de **2011**

Firmado: **Ezequiel Parra Mestre**

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Título del proyecto: Desarrollo del nuevo karat Starter kit	
Autor: Carlos Ming Martínez	**Data:** *Julio de 2011*
Tutor: Jordi Pons Aróztegui	
Titulación: Ingeniería Técnica en Informática de Sistemas	

Palabras clave

- Castellano: Ayuda, formación, demostración, karat.
- Catalán: Ajuda, formació, demostració, karat.
- Inglés: Help, formation, demonstration, karat.

Resumen del proyecto

- Castellano: El proyecto se desarrolla en el marco de un convenio entre la UAB y la empresa UNIT4 en las oficinas de UNIT4 Ibérica en Barberà del Vallès. El objetivo del proyecto es crear una nueva herramienta, denominada *karat Starter Kit Reloaded*, que sirva de autoformación y demostración de la plataforma de desarrollo propia utilizada en la empresa (karat) reforzando los puntos débiles de la anterior y manteniendo los puntos fuertes de los que ya disponía.

- Catalán: El projecte es desenvolupa en marc d'un conveni entre la UAB i l'empresa UNIT4 en les oficines de UNIT4 Ibérica a Barberà del Vallès. L'objectiu del projecte es crear una nova eina, anomenada *karat Starter Kit Reloaded*, d'auto formació i demostració de la plataforma de desenvolupament pròpia utilitzada per l'empresa (karat) reforçant els punts febles de l'anterior i mantenint els punts forts de que ja disponía.

- Inglés: The project is developed under an agreement between the UAB and the company UNIT4 at UNIT4 Iberica's offices at Barberà del Vallès. The project's goal is to create a new self-formation and demonstration tool, called *karat Starter Kit Reloaded*, for karat platform strengthening the weak points of the current one, and maintaining the strengths that it already has.

Tabla de contenido

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

1.1. Marco de trabajo

1.1.1. Convenio de colaboración entre la UAB y UNIT4 Ibérica

Este proyecto se ha desarrollado dentro del marco de un convenio entre la Universitat Autònoma de Barcelona y la empresa UNIT4, en el cual, se estipulan 560 horas para realizar un trabajo de desarrollo de software. Mediante este convenio, el alumno puede adquirir conocimientos de cómo funciona una empresa de desarrollo de software de gestión y de tecnologías de la información, a la vez que este desarrollo sirve como trabajo de final de carrera.

1.1.2. La empresa

UNIT4 tiene una experiencia durante más de cuarenta años en el mundo de la ingeniería del software. Posee un amplio abanico de soluciones de gestión de última generación lo cual la hace ser una de las primeras empresas españolas en tecnologías de la información y comunicaciones.

UNIT4 dispone actualmente de 4.230 trabajadores y cuenta con oficinas y distribuidores en todo el mundo para garantizar un acceso fácil y local a las ventas, servicios y soporte. Se encuentran en: Alemania, Australia, Bélgica, Canadá, Dinamarca, España, Estados Unidos, Estonia, Francia, Holanda, Hungría, Irlanda, Malasia, Noruega, Portugal, Reino Unido, República Checa, Singapur, Suráfrica, Suecia y Uganda.

UNIT4 desarrolla y ofrece la gama de soluciones ekon, tanto ERP como para la gestión del negocio, que se adaptan especialmente a las organizaciones que desarrollan su actividad en un clima de cambios frecuentes y dinámicos (especialmente las centradas en servicios o las del sector público).

UNIT4 Ibérica aplica a sus soluciones la tecnología *karat,* una completa plataforma tecnológica para la gestión de las empresas, que aporta un nuevo concepto de soluciones basado en la independencia total y real de entornos. Su característica principal es que permite la ejecución del software desarrollado con ella en múltiples plataformas tecnológicas, sistemas operativos (MS Windows, Linux, MAC OS) y bases de datos sin modificar el código fuente. Además, ofrece herramientas para la personalización de procesos de negocio en las soluciones, cuya evolución queda garantizada al mismo tiempo que el estándar.

Una de las últimas incorporaciones a karat es *Walnut,* que se trata de la nueva interfaz de usuario que tiene como lema "User experience". Con el fin de optimizar el trabajo cotidiano con la aplicación, centra la innovación en aprovechar la experiencia de los usuarios. Además es mucho más dinámica, personalizable y potencia la operatividad.

Por los requisitos del proyecto éste se sitúa dentro del departamento de *"Research & Development"* que es el grupo que se encarga del desarrollo y el soporte de la herramienta *karat*. Este departamento además se ocupa del desarrollo de nuevas aplicaciones y mantenimiento de las ya existentes.

1.2. Objetivos generales del proyecto

El proyecto tiene como objetivo crear una nueva versión del producto *karat Starter Kit* (KSK en lo sucesivo) cuyo objetivo es la creación de una aplicación guiando al usuario durante el proceso. Desde su lanzamiento inicial se han añadido nuevas funcionalidades a *karat* y el KSK ha quedado desfasado y no cumple al completo con los requisitos de UNIT4.

Con el fin de añadir al KSK las nuevas funcionalidades obtenidas con el paso de *karat* a *Java* y también con el fin de integrarlo al nuevo estilo *Walnut* se decidió crear un nuevo *karat Starter Kit*.

En el capítulo del estudio de viabilidad se explica detalladamente la situación actual del KSK.

A nivel personal este proyecto me proporciona una excelente oportunidad para adquirir experiencia en el sector profesional de la informática, y en este caso en una empresa de ámbito internacional como es UNIT4, y aprender nuevas metodologías de trabajo, antes incluso de terminar mis estudios, ya que se trata de un privilegio del que poca gente puede disfrutar.

Para la realización del proyecto se ha decidido utilizar una de las más novedosas metodologías de gestión de proyectos informáticos, SCRUM, la cual se describe en el siguiente capítulo.

1.3. Introducción a SCRUM

1.3.1. ¿Qué es SCRUM?

Scrum es una metodología iterativa e incremental basada en la filosofía del desarrollo ágil para proyectos, productos y aplicaciones. Esto significa que sus principales características son su agilidad y flexibilidad y sus modelos de equipos de trabajo multifuncionales con capacidad de decisión propia. Sus ciclos cortos de desarrollo se centran en iteraciones rápidas y en la constante opinión del cliente y del equipo de desarrollo.

Scrum tiene una pila de producto, donde están expuestas, por orden de prioridad, todas las funcionalidades del producto. Esta pila se llama *Product backlog.*

Los ciclos de trabajo son denominados *Sprint.* Estos ciclos son iteraciones de 2-4 semanas que se van sucediendo uno tras otro y son de duración fija (terminan una fecha específica y jamás se alargan).

Al principio del *Sprint* el equipo decide qué elementos se comprometen a finalizar del *Product backlog* en el ciclo actual y, una vez empezado, no se pueden cambiar. Esta reunión se llama *Sprint Planning Meeting.*

Diariamente el equipo se reúne para informar del progreso y actualizan unas gráficas sencillas que les orienta sobre el trabajo restante. Dichas reuniones se llaman *Daily Scrum* y cada miembro acostumbra a responder 3 preguntas: ¿Qué hice ayer?, ¿Qué haré hoy? y ¿Qué problemas tengo para poder seguir?

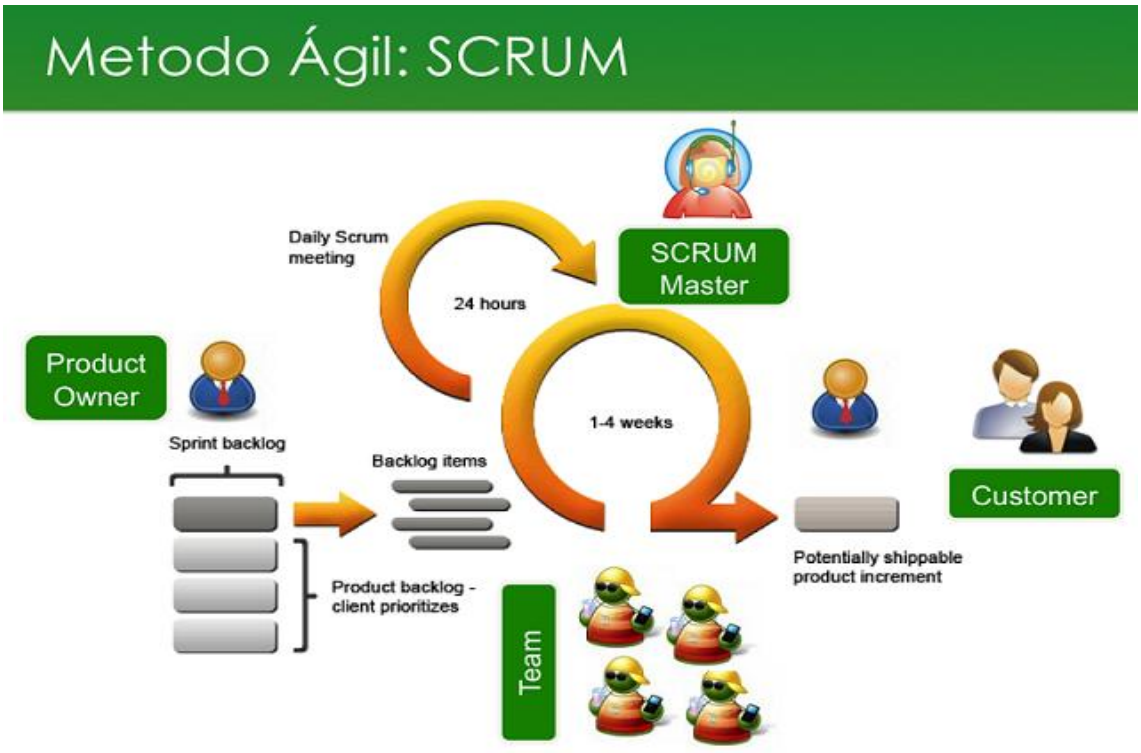


Figura 1: Diagrama de Scrum

Al finalizar el *Sprint* todos los elementos tienen que estar probados y documentados correctamente, y si alguna historia no se ha podido terminar se pasa al siguiente *sprint*. Además el equipo se reúne con los interesados en el proyecto y se les muestra lo que se ha construido. A raíz de estas reuniones se consiguen diferentes comentarios y/o observaciones que se pueden incorporar al *Product backlog* para ser implementados en próximos *Sprints*.

1.3.2. Roles de Scrum

La metodología Scrum requiere de un número de personas que ocupan unos roles específicos:

- Product Owner (PO):

El *Product Owner* es la voz de los clientes. Es la persona que hace de enlace entre el grupo que se encarga del producto y el equipo que lo desarrolla, y asegura que el equipo trabaje de forma adecuada desde la perspectiva de negocio.

El *PO* interpreta y escribe las historias de usuario, las prioriza y las coloca en el *Product Backlog*

- *Scrum* Master (SM):

El *Scrum Master* es el gestor del proyecto, pero no el líder del equipo. Es quien se dedica a quitar impedimentos, proteger el equipo y organiza y modera las reuniones para que el equipo trabaje con el máximo de facilidades posibles para poder alcanzar su objetivo del *Sprint.* A su vez se encarga de mantener la documentación asociada de planificación y control.

- *Equipo de trabajo* (Team):

El *equipo de trabajo* tiene la responsabilidad de crear y entregar el producto. El equipo tiene que estar formado por un pequeño grupo de 5 a 9 personas con las características y habilidades necesarias para poder cumplir los objetivos del *sprint* que ellos mismos seleccionan. Se auto organiza a sí mismo y su trabajo, y hace los problemas visibles.

1.3.3. Documentos de soporte

En Scrum también encontramos un conjunto de documentos propios que han de ser desarrollados durante la realización de la metodología:

- *Product backlog:*

El *Product backlog* es un documento de alto nivel para todo proyecto que proviene de un plan de negocio que puede ser creado juntamente con el cliente, el responsable del cual es el Product Owner, aunque cualquiera puede contribuir. Contiene historias, que son descripciones de todos los requisitos y funcionalidades deseables, priorizadas según su valor para el negocio y estimadas por puntos tanto del valor como del esfuerzo de desarrollo requerido. Esta estimación ayuda al Product Owner a ajustar la línea temporal y la prioridad de las diferentes tareas. Este documento debería ser visible y fácilmente accesible para todo el mundo (especialmente para el equipo).

- *Sprint backlog:*

El *Sprint backlog* es un documento detallado donde se describe cómo el equipo implementará los requisitos y funcionalidades durante el siguiente sprint. Este documento contiene tareas, que son subdivisiones de las historias, estimadas en horas, que no pueden estar valoradas con una duración superior a las 16 horas por tarea. Si una tarea se considera con una duración superior a 16 horas, tendrá que ser dividida con mayor detalle. Las tareas nunca son asignadas directamente, sino que se escogen por los miembros del equipo de la manera que parezca más oportuna, y son actualizadas diariamente.

- *Burndown chart*:

Burndown chart es un gráfico que se muestra públicamente que contabiliza la cantidad de tareas del *sprint backlog* pendientes al comienzo de cada día. Dibujando una línea que conecta los puntos de todos los días pasados del *sprint*, podremos ver el progreso del mismo. Lo normal es que la línea sea descendente hasta llegar al eje horizontal, momento en el que el *sprint* ha terminado. Si durante el proceso se añaden nuevas tareas, la recta

tendrá pendiente ascendente en ciertos momentos, y si se modifican tareas la pendiente variará o hasta puede valer 0 en algunos tramos.

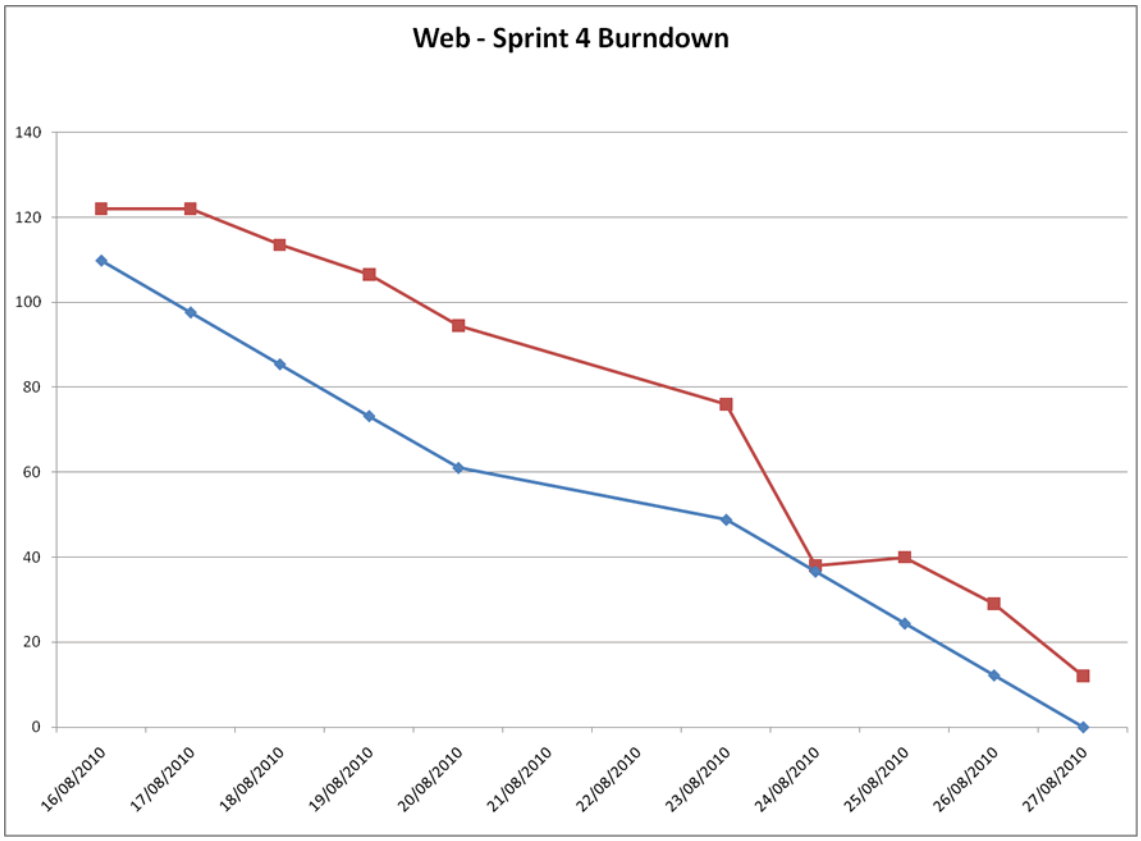


Figura 2: Ejemplo de Burndown chart

1.4. Contenido de la memoria

A continuación se describen los capítulos de los que consta la memoria del proyecto:

- **Introducción:** Apartado en el que se explica brevemente el marco de ubicación del proyecto. Dado que la gestión del proyecto se ha realizado mediante la metodología ágil SCRUM, se da una breve introducción a ésta en dicho apartado.
- **Estudio de viabilidad:** Apartado en el que se realiza una pequeña descripción del sistema actual, objetivos del proyecto, beneficios y riesgos del mismo.
- **Planificación del proyecto:** En este capítulo se muestra cómo se ha gestionado el proyecto mediante un diagrama de Gantt, las fases del mismo y su duración.
- **Análisis de la aplicación:** En este capítulo se analiza como debe ser la aplicación.
- **Entornos de desarrollo:** En este capítulo se detalla el software utilizado para la creación de la aplicación, porque se ha escogido cada entorno de desarrollo y en qué consiste cada uno.
- **Desarrollo de la aplicación:** En este capítulo se detalla en qué ha consistido cada *sprint,* y sus *historias*, así como las pruebas realizadas sobre cada uno para la posterior gestión de los documentos de prueba.
- **Conclusiones:** Capítulo en el que se realiza una valoración personal del alumno sobre el proyecto y las posibles ampliaciones de éste, así como de la experiencia laboral conseguida en UNIT4.

- **Bibliografía:** Detalle de los recursos informativos utilizados para realizar la aplicación.

ESTUDIO DE VIABILIDAD

2.1. ¿Qué es karat Starter kit?

En la actualidad ya se dispone del producto *karat Starter Kit*, *KSK* en lo sucesivo, producto cuyo objetivo es la creación de una sencilla aplicación guiando al usuario durante todo su proceso de construcción. Todos los pasos están documentados e incluyen el código de ejemplo, de manera que permiten un rápido aprendizaje del proceso de desarrollo.

KSK nació con la finalidad de servir como guía de trabajo para todas aquellas personas que se inician en la construcción de aplicaciones mediante la tecnología *karat*, así como para el resto de desarrolladores mediante la inclusión de ejemplos prácticos de nuevas funcionalidades.

2.1.1. Situación actual

El *KSK* actual se basa en el siguiente modelo de datos:

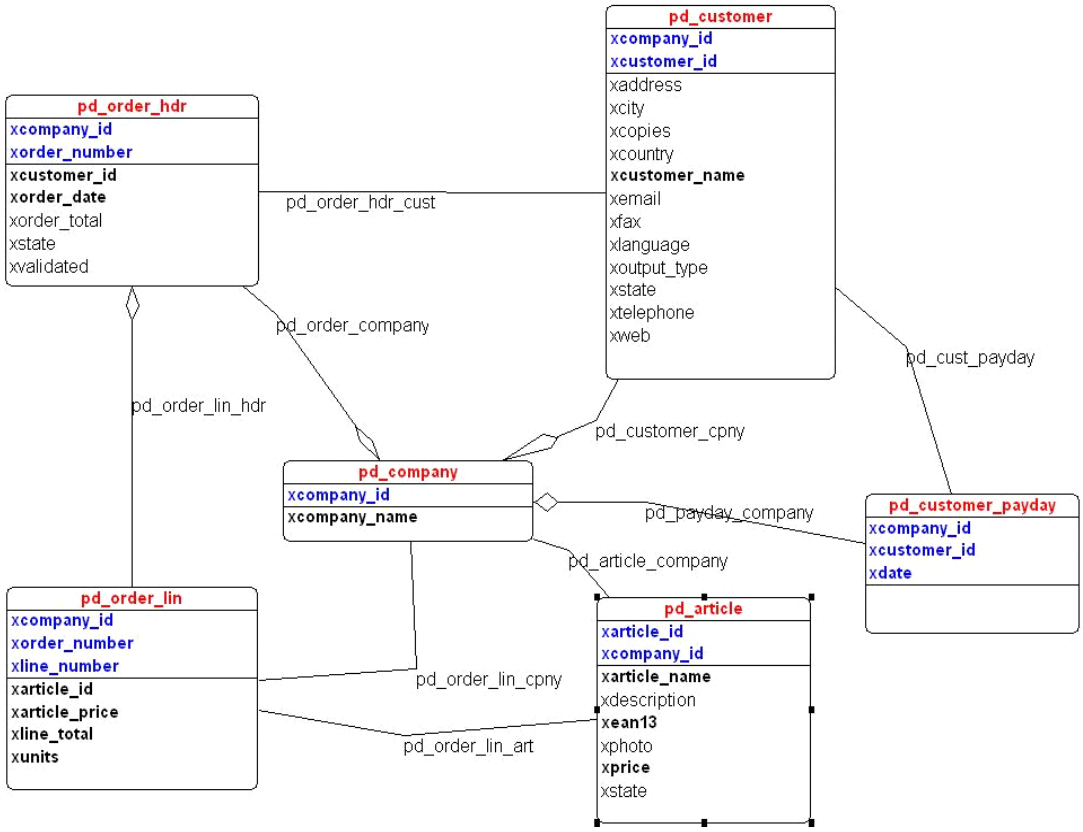


Figura 3: Modelo de datos de karat Starter Kit

Como se puede observar, el modelo de datos pretende simular un entorno de trabajo similar al que usan los clientes a los que está destinado *karat*. Se trata de un modelo de negocio multiempresa (podemos gestionar varias empresas desde una misma aplicación) en los que cada **empresa** dispone de sus propios **clientes**, **artículos** que vender y **pedidos** que gestionar.

KSK está compuesto de varios mantenimientos[1] y cada uno de ellos trata de mostrar la mayor cantidad de funcionalidades de la plataforma.

A continuación se hace una pequeña explicación de cada mantenimiento y de que funcionalidades cumple. En un documento *Anexo* (incluido en el CD) se muestran unas tablas con los mantenimientos y todas las funciones de cada uno.

Módulo de ventas

- Al tratarse de un producto multiempresa se dispone de un formulario de **Selección de empresa activa** que permite seleccionar con que empresa se va a trabajar.
- Mantenimientos de **artículos**: en estos mantenimientos se pueden dar de alta, baja o modificar los artículos de los que dispone la empresa. Desde éstos se pueden realizar además llamadas a otros ordenadores, observar los artículos desde un árbol de navegación, incrementar el precio de los artículos mediante tareas de Workflow (se detalla su funcionalidad en el **Sprint2**) y usar servicios web (se detalla el uso de éstos en el **Sprint3**).
- Mantenimientos de **clientes**: en estos mantenimientos se puede dar de alta, baja o modificar los datos de los clientes. Desde éstos se pueden enviar SMS, realizar un descuento mediante consultas SQL[2] y añadir gráficos de ventas por cliente.
- Mantenimientos de **pedidos**: en estos mantenimientos se pueden realizar, eliminar o modificar pedidos ya existentes. Desde estos mantenimientos se pueden realizar pedidos mediante lenguaje XML[3] o HTML[4] con hipervínculos a los formularios correspondientes e imprimirlos con un formato predefinido (se detalla esta funcionalidad en el **Sprint1**).

Módulo de enseñanza

- En estos mantenimientos se puede gestionar cursos, exámenes y alumnos para mostrar la herramienta de evaluación que se trata de una creación de vistas dinámica.

Módulo de ejemplos

[1] *Mantenimiento*: Son pantallas que representan gráficamente un objeto de negocio. Normalmente se utilizan para realizar el mantenimiento de datos del objeto de negocio (alta, baja, modificación).
[2] *SQL*: lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas.
[3] *XML*: es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos
[4] *HTML*: lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes

- En estos mantenimientos se muestran ejemplos varios de funcionalidades que no se han podido incorporar al modelo de negocio anterior. Forman parte de este módulo varios mantenimientos sin relación que muestran funciones tales como:
    - Mostrar el uso de scripts ubicados en el equipo del cliente.
    - Generación de árboles.
    - Gestor de imágenes.
    - Crear gráficos de varios tipos usando un fichero XML para ello.
    - Apertura de formularios ubicados en otros servidores y otra configuración.
    - Creación y programación de diagramas de Gantt.

A medida que se han sucedido las actualizaciones de karat, este modelo de datos no ha podido reflejar todas las funcionalidades de la herramienta y se han ido añadiendo mantenimientos y nuevas funciones a formularios ya existentes para poder ofrecer una visión completa de karat.

A continuación se muestran unas capturas de pantalla de estos casos:

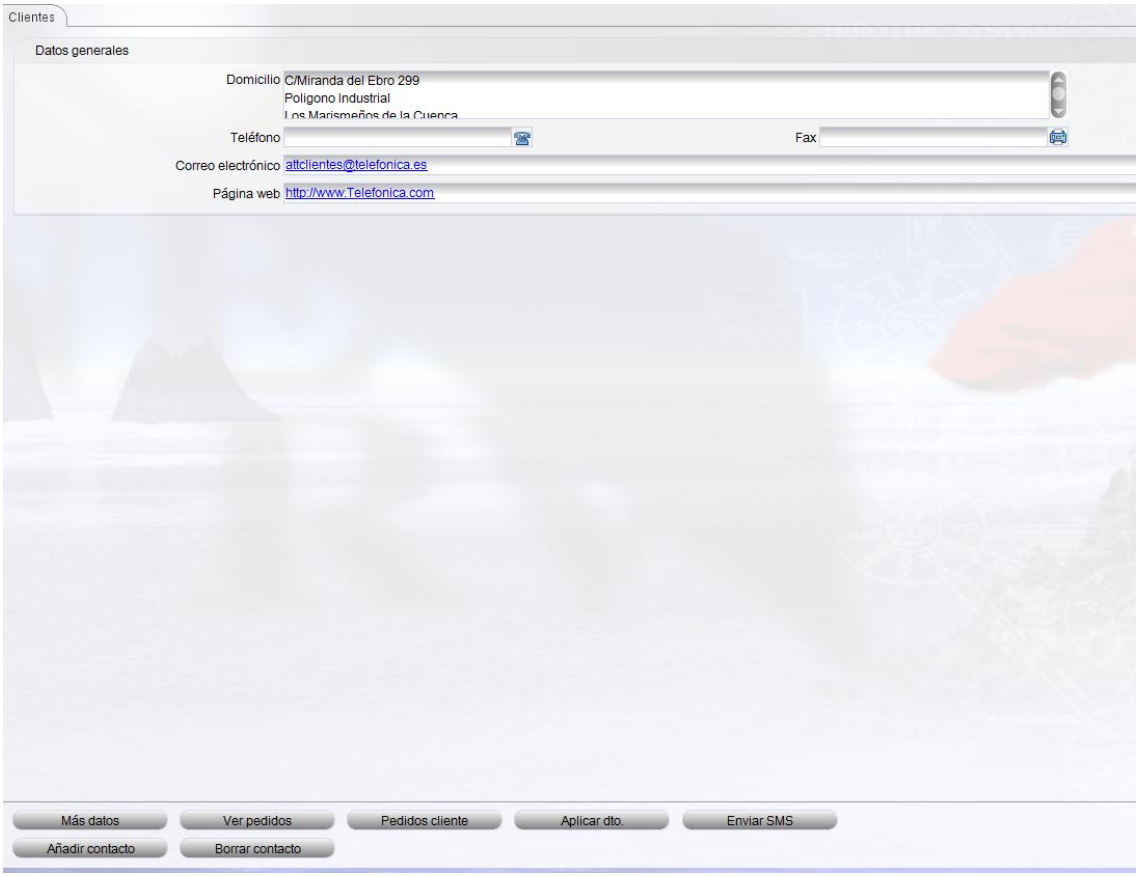


Figura 4: Formulario con sobrecarga de botones

Como se puede observar en la imagen hay formularios con hasta 6 botones lo cual produce una carga visual alta y muestran un formulario poco atractivo a la vista.

En cuanto a la nomenclatura usada para las estructura interna del *KSK* tampoco se ha seguido un criterio uniforme lo cual lleva a que se puedan encontrar varias nomenclaturas distintas en objetos de un mismo grupo.

A continuación se muestra un ejemplo de este caso:

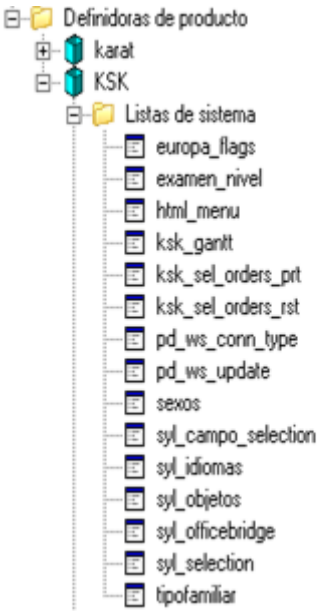


Figura 5: Nomenclatura de las listas del sistema

Como se puede observar no se sigue una nomenclatura estándar en listados que forman parte del mismo *KSK*, los hay que empiezan por KSK_, pd_, syl_…

2.1.2. Diagnóstico

El *KSK* actual es una primera herramienta para realizar pruebas y/o demostraciones de todas las funcionalidades *karat*¸ pero como ya se ha dicho, *KSK* constaba de un modelo inicial al que se han ido añadiendo nuevas funcionalidades, llegando al punto de que algunas de éstas aparecen de forma poco atractiva o un poco forzada y que a la vez dan la impresión de seguir unos criterios no unificados, además de no seguir estrictamente los nuevos criterios *Walnut.*

2.1.3. Conclusiones

El objetivo que se persigue ahora es disponer de una nueva versión del producto *KSK* totalmente remozada, reforzando sus fortalezas pero reduciendo las debilidades que presenta, *KSK Reloaded*.

El producto *KSK* debe ser lo suficientemente ágil para que pueda ser útil para diferentes tipos de usuarios, por ejemplo como instrumento de formación y autoformación, para el

departamento comercial en las demostraciones de la plataforma *karat*, para fábrica en los procesos de prueba, etc.

2.2. Objetivos del proyecto

El objetivo propuesto para este proyecto es que *KSK* no sólo debe tener una gran funcionalidad sino que también tenga las siguientes características:

1. Producto cercano a los productos del portafolio de Unit4; por ejemplo un módulo de ventas en el que intervengan empresas, clientes, artículos, pedidos, etc.
2. Debe tener una alta capacidad para ser entendido, aprendido, usado y ser muy atractivo para el usuario.
3. Debe incorporar las novedades de la plataforma para que puedan ser mostradas de forma natural y atractiva.
4. Modelo de datos simple y claro para que sea fácilmente entendido.
5. Interfaz de usuario siguiendo el nuevo estilo *Walnut*.
6. Producto multiidioma. Idiomas contemplados español e inglés.
7. Excelente ayuda integrada que indique el qué, pasos y resultados.

	Critico	Prioritario	Secundario
O1	X		
O2		X	
O3	X		
O4	X		
O5	X		
O6		X	
O7			X

Tabla 1: Prioridades de los objetivos

2.2.1. Partes interesadas

A continuación presentemos las partes interesadas del proyecto.

Nombre	Descripción	Responsabilidad
Depto. "*Research & Development*" UNIT4	Departamento de la empresa UNIT4 dedicado a la parte de desarrollo y mantenimiento del software.	Departamento encargado de desarrollar y mantener nuevos productos de software.

Tabla 2: Partes interesadas

2.2.2. Equipo del proyecto

A continuación se muestra una tabla reflejando cómo se ha confeccionado el equipo del proyecto y una pequeña explicación de la responsabilidad de cada uno.

Nombre	Descripción	Responsabilidad
Jordi Pons Aróztegui	Tutor del Proyecto(DP)	Supervisa el trabajo del alumno durante el proyecto.
Ezequiel Parra Mestre	Jefe de Proyecto UNIT4 (CP)	Define, gestiona y controla el proyecto.
Carlos Ming Martínez	Analista (A)	Desarrolla el estudio de viabilidad y la planificación. Análisis de la aplicación: arquitectura, metodología, especificación, estándares... Participa en el diseño y validación.
Eduard Mestre	Analista (A)	Asesoramiento en el análisis de la aplicación y asesoramiento técnico a lo largo del proyecto.
Carlos Ming Martínez	Programador (P)	Desarrolla la aplicación de acuerdo al análisis y la planificación prevista. Hace la implantación del proyecto.
Carlos Ming Martínez	Diseñador (D)	Diseña y desarrolla el aspecto de la aplicación de acuerdo al análisis y normas de usabilidad establecidas.

Tabla 3: Equipo del proyecto

2.3. Requisitos del proyecto

2.3.1. Requisitos funcionales

A continuación tenemos la lista de los requisitos funcionales que tendrá que cumplir el software que se va a crear.

RF1. Creación de un módulo de ventas.
RF2. Gestión de los objetos de negocio.
RF3. Creación de los formularios.
RF4. Gestión de impresos.
RF5. Definir un Modelo Organizacional
RF6. Incorporar Simple Workflow.
RF7. Incorporar escritorios, cajas y atributos
RF8. Incorporar Web Services.
RF9. Incorporar cuadro de mando, uso de servicio, Widgets y Workspaces.
RF10. Incorporar un ejemplo sencillo de karat Office Bridge.
RF11. Despliegue del nuevo producto y hacer una copia de seguridad de los códigos Java.
RF12. Documentación de usuario del nuevo producto.

2.3.2. Requisitos no funcionales

RNF1. Cumplimiento de la guía de estilo *Java* para *karat*.
RNF2. Cumplimiento de la guía de estilo Walnut.
RNF3. Control de todas las entradas de usuarios.
RNF4. Tolerancia a errores y acciones incorrectas.
RNF5. Todos los menús deben ser multiidioma, español e inglés.

2.3.3. Restricciones del sistema

1. Seguir los estándares de UNIT4.
2. La aplicación debe adaptarse al sistema físico disponible en la entidad.
3. El proyecto debe finalizarse antes del 20 de junio de 2011.

2.3.4. Catalogación y priorización de los objetivos

#REQ	Esencial	Condicional	Opcional
RF1	X		
RF2	X		
RF3	X		
RF4	X		
RF5		X	
RF6	X		
RF7		X	
RF8	X		
RF9		X	
RF10	X		
RF11	X		
RF12			X
RNF1	X		
RNF2	X		
RNF3		X	
RNF4		X	
RNF5			X

Tabla 4: Catalogación y priorización de los objetivos

2.4. Alternativas y selección de la solución

2.4.1. Alternativa 1

Dado que actualmente ya se dispone de un *KSK*, se puede considerar el hecho de aprovechar al máximo éste y modificarlo para que se integre en el estilo *Walnut* y conseguir que siga estrictamente las guías de estilo *Walnut* y *Java*.

Si se considera esta alternativa se tiene que tener en cuenta que se debe modificar el modelo de datos, por uno que muestre todas las nuevas funcionalidades de *karat* con todo lo que ello implica: rehacer y renombrar las tablas del repositorio, las consultas base, objetos de negocio, formularios y listados; y aprovechar que hay que modificarlos para aplicarles un nuevo estándar en la nomenclatura.

Dado que el *KSK* actual, no sigue un estándar en la nomenclatura de las tablas ni en todo lo que deriva de ellas, modificarlo todo podría suponer una inversión crítica de tiempo.

2.4.2. Alternativa 2

En este caso se opta por olvidar completamente el *KSK* actual y realizar uno totalmente nuevo con un modelo de datos nuevo, y creándolo desde sus cimientos con una nomenclatura que siga el estilo *Walnut* y que cumpla las normas de nomenclatura *Java* para *karat*. De este modo se invierte algo más de tiempo en el diseño inicial pero se evitan los errores derivados de una adaptación, ya que con unas bases sólidas lo único que hay que hacer es ir subiendo nivel a nivel conservando el estilo marcado.

2.4.3 Comparación de las alternativas propuestas

Alternativa	Costes de Adquisición	Costes de adaptación	Soporte	Nivel de integración	Complejidad	Formación
1	0€	Alto	Incluido en el proyecto	Bajo	Alta	Incluida en el proyecto
2	0€	Bajo	Incluido en el proyecto	Alto	Media	Incluida en el proyecto

Tabla 5: Comparación de las alternativas propuestas

2.4.4. Solución propuesta

Para este proyecto se ha tomado la decisión de seguir la alternativa 2, es decir, realizar el nuevo *KSK* desde cero pero aprovechando el conocimiento de la primera versión, ya que se adapta mejor al cliente y se ahorra complejidad al tratar de adaptar el producto anterior.

2.5. Planificación del proyecto.

2.5.1 Recursos del proyecto

En este apartado se muestra lo referente a la planificación del proyecto y los recursos de qué se disponen para realizarlo.

El coste de los recursos humanos es aproximado, ya que por este proyecto el alumno recibirá un importe de 2.352€. El coste ficticio del proyecto supondría alrededor de unos 22.280€, como se detalla en el apartado 2.6 de presupuesto.

- Recursos humanos:

Función	Coste/h
Jefe de proyecto	100€/h
Analista	50€/h
Programador	30€/h
Técnico de pruebas	20€/h

Tabla 6: Recursos humanos del proyecto

- Recursos materiales:
  - Licencia karat.
  - Equipo:
    PC – Intel Core 2Q8400 @ 2.66GHz
    6,00 GB RAM
    Disco duro 500GB
    Conexión a internet
  - Software:
    Windows 7
    Eclipse IDE for Java EE Developers
    karat Studio
    karat Escritorio
    Microsoft FrontPage
    Unit4Help

2.5.2. Planificación temporal

El proyecto se desarrollará de Noviembre de 2010 hasta Junio de 2011, con una dedicación de aproximadamente 20h semanales y un total de 560h.

Fecha de inicio: 8 de noviembre de 2010

Fecha de finalización: 17 de junio de 2011

La planificación es muy susceptible a variar ya que se han planificado jornadas laborales de 4h, sin tener en cuenta posibles ausencias por exámenes u otros tipos de ausencias justificadas.

Todas las fases del proyecto se desarrollan mediante un modelo lineal. Por lo tanto, cada fase no comienza hasta que no se ha terminado la fase anterior.

A partir de principios de enero se pasará a utilizar la metodología ágil Scrum con lo cual cada Sprint conlleva en sí mismo una fase de análisis, diseño, desarrollo y pruebas.

A continuación se muestra una tabla con las tareas a desarrollar durante el proyecto y su correspondiente Diagrama de Gantt.

Tarea	Nombre de tarea	Función	Predecesoras
1	**karat Starter kit**	**149 días**	
2	Curso de formación	10 días	
3	**Planificación**	**6 días**	**2**
4	Estudio de viabilidad	2 días	
5	Aprobación del estudio de viabilidad	1 día	4
6	Plan de proyecto	2 días	5
7	Aprobación del plan de proyecto	1 día	6
8	**Análisis de la aplicación**	**12 días**	**3**
9	Reunión con el departamento de I+D	1 día	
10	Análisis de requisitos	2 días	9
11	Documento de requisitos	1 día	10
12	Investigación acerca d la situación actual	5 días	11
13	Documento de análisis	2 días	12
14	Aprobación del análisis	1 día	13

Tarea	Nombre de tarea	Función	Predecesoras
15	**Presentación del proyecto**	**19 días**	**8**
16	Diseño del módulo básico	5 días	14
17	Desarrollo del módulo básico	8 días	16
18	Preparación de la documentación	3 días	17
19	Preparar demostración	3 días	18
20	**Sprint 0**	**14 días**	**15**
21	Sprint Planning meeting	1 día	19
22	Definir tablas, consultas y objetos de negocio	5 días	21
23	Configuración de la herramienta	3 días	22
24	Consolidar objetos de negocio y formularios	5 días	22
25	**Sprint 1**	**15 días**	**20**
26	El listado de artículos con código de barras.	4 días	24
27	Consolidar los listados y los formularios de KSK Reloaded.	2 días	26
28	Incorporar el uso de MO. Roles y usuarios.	1 día	27
29	Formación SW y ver el uso actual en el KSK.	2 días	28
30	Una regla de negocio para validar un pedido.	1 día	29
31	Una regla de tiempo para la verificación diaria del stock.	1 día	30
32	Un diálogo de validación de stock con llamada a programa y seguimiento.	2 días	31
33	BUGS Sprint 1	1 día	32
34	Reuniones periódicas Sprint 1	1 día	33
35	Traspaso de datos KSK a KSK2	1 día	32
36	**Sprint 2**	**16 días**	**25**
37	Uso de atributos de BO en KSK	1 día	35
38	Formación en escritorios y cajas y ver su uso actual en KSK.	1 día	37
39	Un escritorio con pedidos por cliente y otros...	5 días	38
40	Formación WS y ver uso actual en el KSK.	2 días	39
41	Incorporar WS a KSK2.	5 días	40
42	BUGS Sprint 2	1 día	41
43	Reuniones periódicas de Sprint 2	1 día	42
44	**Sprint 3**	**17 días**	**36**
45	Completar formulario que vía WS muestre los datos de un artículo	4 días	43
46	Formación acerca de cuadro de mando y ver el uso actual en el KSK	1 día	45
47	Añadir cuadro de mando, uso de servicio, Widgets y Workspaces	5 días	46
48	Incorporar ejemplo sencillo de uso de KOB	3 días	47
49	Incorporar búsqueda rápida	1 día	48
50	Despliegue del nuevo producto	2 días	49
51	Reuniones periódicas Sprint 3	1 día	50
52	**Sprint 4**	**18 días**	**44**
53	Documentación de usuario del nuevo producto	15 días	51
54	Despliegue del nuevo producto	2 días	53
55	Reuniones periódicas Sprint 4	1 día	54
56	**Generación de la Documentación**	**10 días**	52
57	**Presentación del proyecto en UNIT4**	**9 días**	**56**
58	Preparación de la demostración	5 días	56
59	Preparar power point	3 días	58
60	Presentación en el aula de conferencias	1 día	59
61	**Cierre del proyecto**	**1 día**	57
62	**Defensa del Proyecto**	**1 día**	61

Tabla 7: Tareas del proyecto

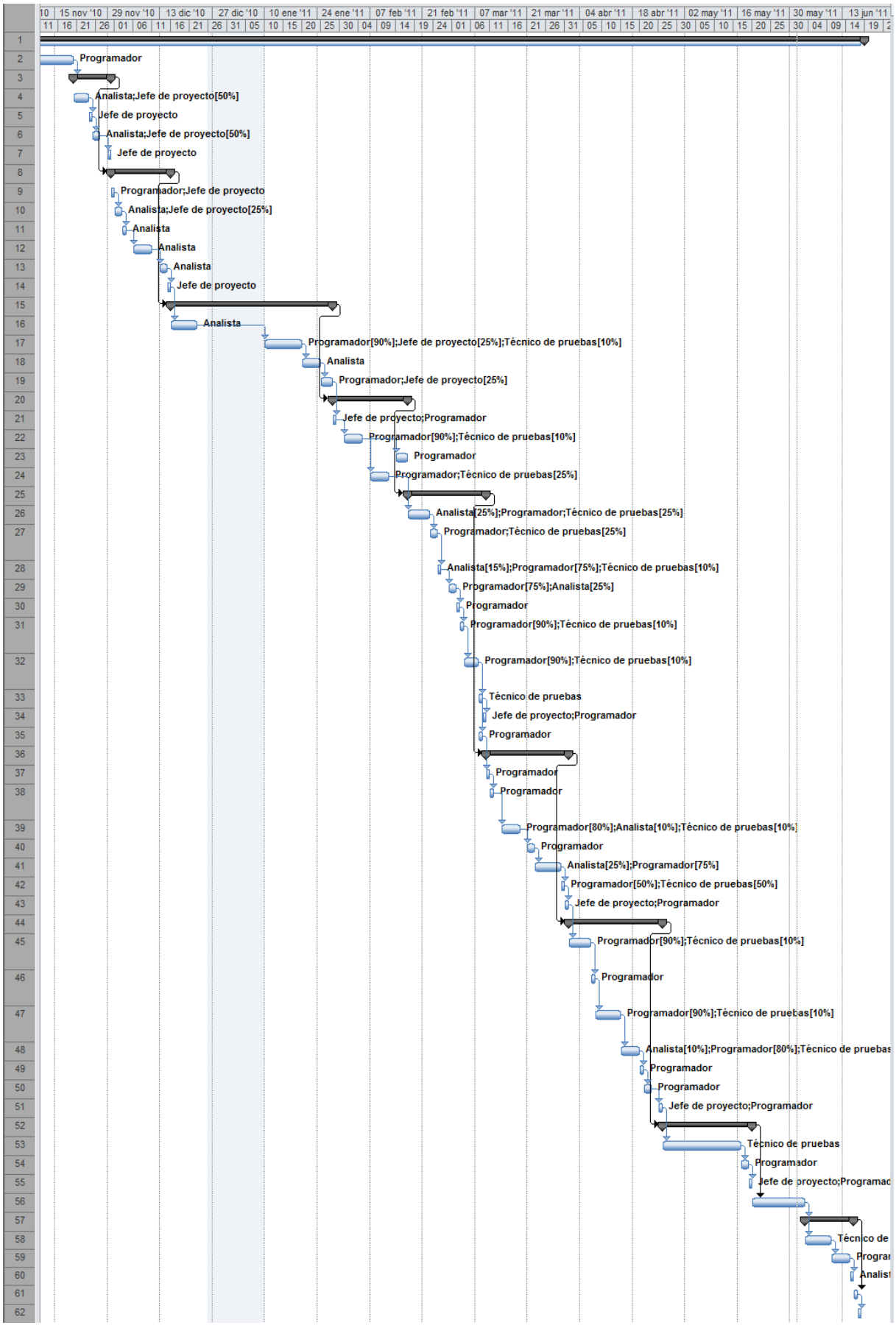


Figura 6: Diagrama de Gantt

2.6. Presupuesto

2.6.1. Estimación del coste de personal

A continuación se muestra una tabla con el coste personal de los recursos del proyecto.

Función	Coste/h	Horas	Coste total
Jefe de proyecto	100€/h	57	5700€
Analista	50€/h	95	4750€
Programador	30€/h	321	9630€
Técnico de pruebas	20€/h	123	2460€
TOTAL		**596**	**22540€**

Tabla 8: Coste personal del proyecto

2.6.2. Estimación de coste de los recursos

A continuación se muestra una tabla con el coste de los recursos utilizados para realizar el proyecto.

Recurso	Coste
Ordenador personal	600€

Tabla 9: Coste de los recursos del proyecto

Los programas usados para la realización del proyecto son o bien de código abierto, propiedad de UNIT4 o se han realizado con licencias de las que ya dispone la empresa (caso de Microsoft Office) con lo que no conllevan un coste adicional.

2.6.3. Resumen y análisis de coste beneficio

A continuación se muestra una tabla resumen con los costes totales del proyecto.

Recurso	Coste
Coste de personal	22540€
Coste de recursos	600€
Total	**23140€**

Tabla 10: Resumen de costes del proyecto

El coste del proyecto sería relativamente elevado, pero teniendo en cuenta que es un producto que se incorporará en próximas versiones de karat, y que éste será lo primero que vean los futuros clientes potenciales se puede amortizar rápidamente si se consiguen los objetivos fijados inicialmente y el nuevo KSK consigue atraer a clientes indecisos.

2.7. Conclusiones

A raíz del estudio de todos los datos mostrados se puede llegar a la conclusión que el proyecto a realizar goza de beneficios e inconvenientes (característica propia de todo proyecto) y además se pueden enumerar los beneficios que obtiene la empresa con él.

Beneficios:

- Modelo simple
- Diseño atractivo para el usuario.
- Incorporación de novedades.
- Interfaz de usuario al estilo *Walnut.*
- Producto multiidioma.
- Excelente ayuda integrada que indique el qué, pasos y resultados.
- Mejora en la formación de nuevos usuarios.

Inconvenientes:

- Recelo por parte de los usuarios.
- Necesidad de un periodo de adaptación al nuevo modelo de datos.

Los beneficios y el estudio de viabilidad realizado nos llevan a la conclusión de que el proyecto es viable.

ANÁLISIS DE REQUISITOS

3.1. Perfiles de usuario

A la aplicación pueden acceder todos los usuarios de karat

- PU. Usuario karat: Cualquier usuario con licencia karat puede acceder a esta aplicación e interactuar con todas sus funcionalidades.

3.2. Requisitos funcionales

A continuación se detallan los requisitos funcionales que debe cumplir la aplicación:

- RF1. Creación de un módulo de ventas: El producto a desarrollar debe ser un producto cercano al portafolio de Unit4. Se ha decidido crear un módulo de ventas similar al que se disponía anteriormente con empresas, clientes, artículos y pedidos. No obstante se han añadido campos a los formularios para aumentar su funcionalidad, hacerlo más actual y más práctico para el usuario.
- RF2. Gestión de los objetos de negocio: La estructura básica del proyecto se debe realizar mediante la herramienta karat denominada *objeto de negocio* (se explica detalladamente su estructura y funcionamiento en el capítulo de entornos de desarrollo) con los que se debe especificar y restringir los tipos de datos que acepta cada campo.
- RF3. Creación de los formularios: La presentación de los datos se realiza mediante la herramienta karat denominada *formulario* (se explica detalladamente su estructura y funcionamiento en el capítulo entornos de desarrollo). A través de éstos se presenta la información de forma ordenada y el usuario interactúa con la información de la base de datos ya sea para consulta, creación o modificación. Según el tipo de formulario del que se trate se muestra una información u otra.
- RF4. Gestión de impresos: La impresión de los datos se realiza mediante la herramienta karat denominada *listado* (se explica detalladamente su estructura y funcionamiento en el capítulo de entornos de desarrollo). Algunos formularios deben contar con la posibilidad de ser impresos con un formato prediseñado, uniforme a la vez que claro y atractivo.
- RF5. Definir un Modelo Organizacional: Debe mostrarse la funcionalidad de *modelo organizacional* ofrecida por karat como una funcionalidad importante de la que puede derivar el control de acceso de usuarios así como si disponen de algún rol determinado en la empresa. Es clave definir este modelo ya que a raíz de esta estructura de organización derivan aspectos de otras funcionalidades que se definirán más adelante.
- RF6. Incorporar Simple Workflow: Debe incorporarse el uso de *Simple Workflow* (se explica detalladamente su estructura y funcionamiento en el Sprint 1) a la aplicación. Se deben crear varios procesos automatizados, uno que avise al responsable en caso de que un pedido exceda un cierto importe y otro que verifique el stock de los artículos ya sea cada día automáticamente como a través de una llamada y con la visualización del estado de la ejecución.

- RF7. Incorporar Escritorios, cajas y atributos: Se debe mostrar la nueva funcionalidad de esta versión de karat mediante el ejemplo de un *Escritorio* sencillo adaptando el contenido del KSK para ser visualizado mediante este nuevo estilo diseñando varias cajas y relacionando los atributos de cada una de tal manera que se muestre la conexión e interacción entre ellas.
- RF8. Incorporar Web Services: Se debe incorporar un ejemplo sencillo de Web Services mediante el que se muestre la carga de datos de un formulario karat mediante el uso de servicios web sin necesidad de usar consultas a la base de datos.
- RF9. Incorporar cuadro de mando, uso de servicio, Widgets y Workspaces: La aplicación debe contener un conjunto de *Widgets* (se explican en el Sprint 3) para la monitorización del módulo de ventas y que sea de interés para el usuario y crear un *Workspace* con todos ellos para permitir cargarlos a la vez.
- RF10. Incorporar un ejemplo sencillo de karat Office Bridge: Se debe incorporar un ejemplo sencillo que muestre como se generan documentos de texto y hojas de cálculo de forma totalmente transparente para el usuario y gestionados por la propia aplicación, de manera que se muestren siguiendo unas plantillas predefinidas o bien en un documento plano los datos más relevantes de un pedido.
- RF11. Despliegue del nuevo producto y hacer una copia de seguridad de los códigos Java: Crear un ejecutable que permita la instalación del producto íntegramente, incluyendo plantillas necesarias, de manera que se pueda instalar en cualquier equipo con karat y realizar una copia de seguridad de los códigos Java utilizados para la implementación de la aplicación.
- RF12. Documentación de usuario del nuevo producto: Crear un ejemplo de nueva documentación de usuario que indique “El qué, cómo, y que pasos hay que seguir” que sirva como autoformación para aprender cómo crear un módulo nuevo e independiente de negocio y/o para consolidar los conocimientos sobre todas las funcionalidades mostradas en KSK.

3.3. Requisitos no funcionales

A continuación se muestran los requisitos no funcionales que debe cumplir la aplicación.

- RNF1. Cumplimiento de la guía de estilo *Java* para *karat*: Se debe seguir estrictamente la guía de estilo *Java* estipulada para la plataforma karat en cuanto a los criterios a seguir para nombrar funciones, variables y otros objetos necesarios para la codificación de la aplicación. Para tal tarea se consultarán las reglas contenidas en el documento interno: *pqGuíaEstiloProgramaciónJava.odt*.
- RNF2. Cumplimiento de la guía de estilo Walnut: Se debe seguir estrictamente la guía de estilo *Walnut* para el diseño de los formularios de la aplicación, tanto de los criterios visuales como de los internos que no son visibles para el usuario común pero si para usuarios con ciertas licencias más avanzadas.
- RNF3. Control de todas las entradas de usuarios: La aplicación debe controlar el acceso de los usuarios así como los permisos de que disponga cada uno.

- RNF4. Tolerancia a errores y acciones incorrectas: La aplicación debe permanecer estable aun cuando el comportamiento del usuario no sea el esperado, debe mantenerse en funcionamiento y gestionar correctamente que se escriba el tipo de datos esperado (numérico, alfanumérico….) en cada casilla de los formularios.
- RNF5. Todos los menús deben ser multiidioma, español e inglés: La aplicación se debe desarrollar en dos idiomas: español e inglés. Todos los datos que se muestren por pantalla deben ser visibles en cualquiera de los dos idiomas en función de la configuración escogida por el usuario.

ENTORNOS DE DESARROLLO

4.1. karat

Para la realización del proyecto se ha usado la tecnología karat explicada brevemente en la introducción. En este capítulo se detallan las características de karat, la estructura de las aplicaciones creadas con esta plataforma y su funcionalidad.

karat garantiza la máxima independencia respecto a los elementos tecnológicos (bases de datos, sistemas operativos, terminales, etc.) y facilita un alto grado de autonomía en la automatización de los procesos de negocio gracias a su completo abanico de asistentes, que simplifican al usuario tareas que antes realizaban consultoras.

La plataforma tecnológica karat ofrece la máxima agilidad de uso tras la implementación, y además permite que los cambios se produzcan de forma rápida, instantánea y casi sin coste. Junto a todo ello, facilita al máximo la personalización de la gestión y consigue reducir a la mínima expresión los costes de propiedad.

El resultado final son aplicaciones de gestión totalmente flexibles y con una tecnología tan avanzada que pueden personalizarse sin necesidad de tocar fuentes. Además, permiten que los upgrades / actualizaciones se realicen de forma inmediata, con todas las personalizaciones y prácticamente sin costes.

karat también permite a los usuarios la creación de indicadores dinámicos (Widgets) asociados a numerosos conceptos de negocio. Estos indicadores permiten, desde un dispositivo móvil o un ordenador, acceder instantáneamente y con un solo clic a una gran cantidad de información de UNIT4 ekon o internet, y generan avisos cuando se producen algunas situaciones. Fácilmente personalizables, pueden ser agrupados en cuadros de mando (Workspaces) a la medida de cada usuario. De este modo se optimiza la funcionalidad y apariencia de las aplicaciones

Toda aplicación karat se caracteriza por tener una estructura concreta. Se trata de una serie de elementos entrelazados entre sí y que le dan la versatilidad y flexibilidad deseada.

A continuación se muestra una figura con la estructura principal de las aplicaciones karat:

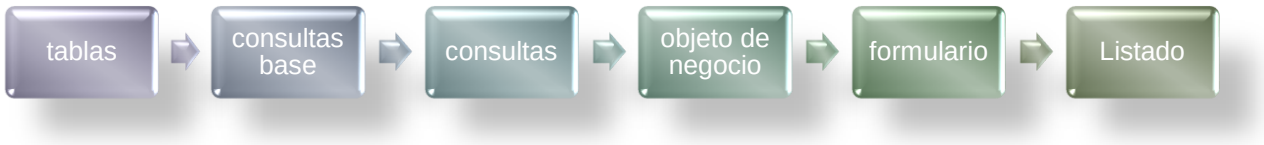


Figura 7: Estructura de las aplicaciones karat

En los apartados siguientes se explica detalladamente qué es cada estructura, sus características y qué función tienen.

4.1.1. Herramientas básicas: variables de entorno, dominios y contadores

Las herramientas que se describen a continuación no se han mostrado en la figura anterior ya que, como tales, no forman parte importante de la estructura de una aplicación karat, pero se trata de herramientas que facilitan la gestión de algunas estructuras y en otros casos automatizan procesos que requerirían de varias líneas de código para implementarlas.

- Variables de entorno:

Como el nombre indica se trata de variables, pero en este caso no es ningún dato perteneciente a una tabla, sino variables que se guardan directamente en la base de datos y que se acostumbran a utilizar para aplicar algún tipo de filtro a los datos de nuestras tablas. Se puede escoger la opción de actualizar el valor de esta variable cada vez que se modifique desde un formulario.

- Dominios:

Podríamos definir un dominio como una plantilla para un campo de una tabla. Al definir un dominio establecemos varios parámetros con un valor determinado. A continuación se muestran los parámetros que podemos asignar:

  - Tipo de datos: entero, fecha, booleano, texto….
  - Valor por defecto: se puede asignar si se desea un valor por defecto.
  - Obligatorio: convertir en obligatorio este campo.
  - Características del control:
    - Tipo de la caja que contiene el dato en el formulario.
    - Dimensiones de la caja.
    - Mensaje de ayuda.
    - Alineamiento del texto.
  - Etiquetas de entrada/salida: aquí se define el texto que aparecerá en el formulario para identificar este control.
  - Máscara de entrada: aquí se define el tipo de caracteres que se permite introducir mediante símbolos, ej.: ###### para aceptar seis caracteres numéricos.
  - Máscara de salida: en este campo se define el tipo de caracteres que se mostrará por pantalla mediante símbolos, ej.: ###### para mostrar hasta seis caracteres numéricos.

- Contadores:

Esta herramienta, como el nombre dice, nos sirve para crear un contador que luego se puede hacer servir en varios formularios para distintas enumeraciones. Para cada uno de ellos se puede definir un valor inicial y un incremento individual, según se desee que incremente de uno en uno como con un intervalo concreto

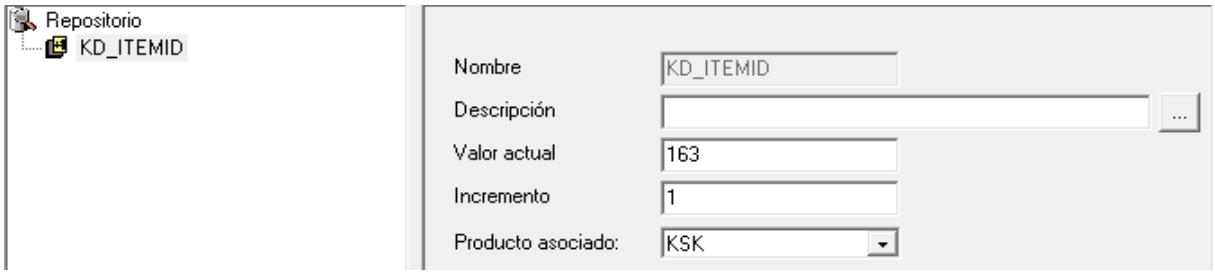


Figura 8: Contador

4.1.2. Diseño de la base de datos: Tablas, consultas y consultas base

En este apartado se explica el diseño de las tablas, consultas y consultas base que son el motor principal de la aplicación.

- Tablas:

  Las tablas son los elementos que contienen todos los datos que se recogen a través del programa. Están formadas por filas y columnas donde cada fila contiene todos los datos de una entrada y cada columna contiene un tipo de dato distinto. El tipo de datos, denominado campo, se define mediante la definidora de tablas de karat.

  A través de la definidora especificamos todas las características de cada campo. Si se desea se puede usar un dominio para establecer todos los datos de un campo en concreto o al menos los datos más generales dejando sólo para concretar los específicos. Es en este punto en el que tener varios dominios definidos acelera cuantiosamente la creación de tablas. Se omite la enumeración de todos los datos a rellenar en las tablas ya que son los mismos que los mencionados en el apartado anterior para la creación de los dominios.

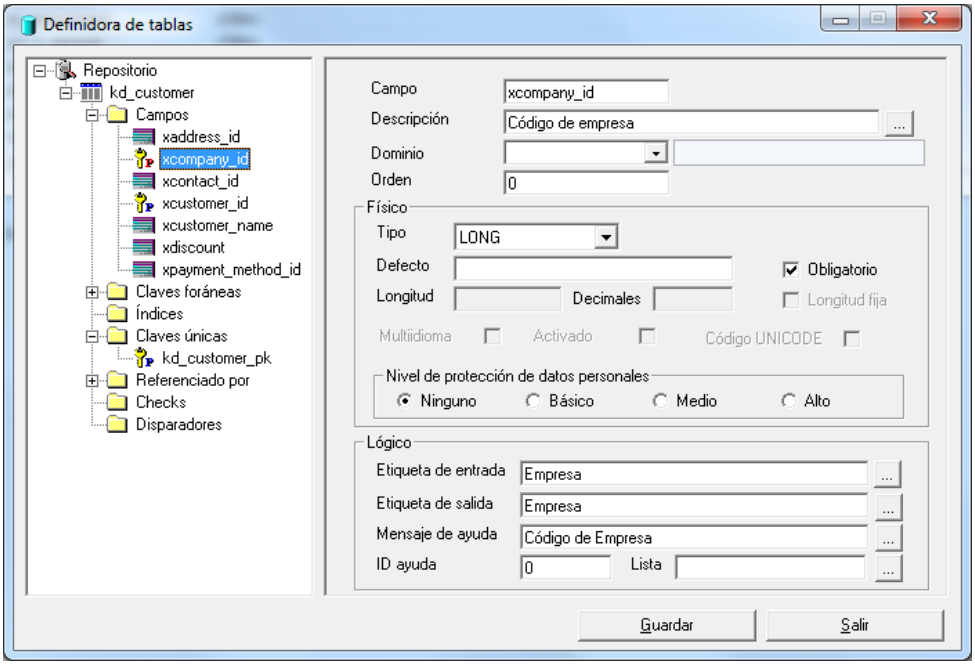


Figura 9: Definidora de Tablas

- Consultas base:

  Las consultas base son sentencias SQL mediante las que obtenemos los datos de una tabla para poder recuperarlos. Generalmente las consultas base son simplemente una consulta a **todos** los campos que contiene una tabla para después procesarlos como sea conveniente. No obstante, mediante la definidora de consultas base se puede seleccionar qué campos se desea obtener, si se desea ordenarlos con algún criterio y también si se desea poner algún tipo de restricción respecto a un campo concreto. De todos modos, por lo general, las consultas base acostumbran a ser una consulta a todos los datos de la tabla.

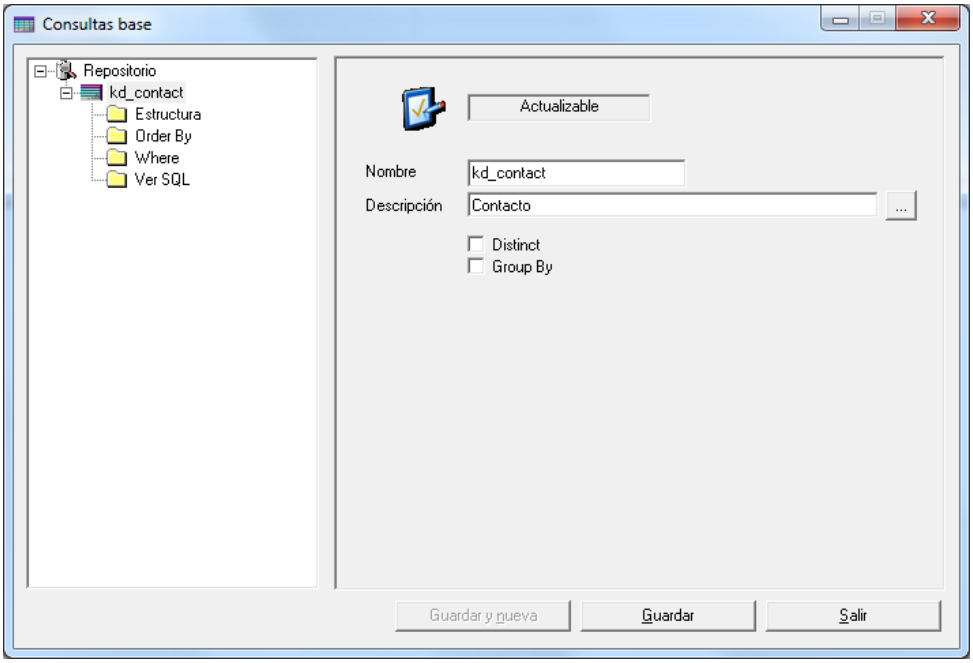


Figura 10: Consultas base

- Consultas:

  Éstas son muy parecidas a las consultas base, de hecho, cada vez que se genera una consulta base con la definidora de consultas base se genera automáticamente una consulta exactamente igual.

  La diferencia que hay entre las consultas base y las consultas es que en las consultas base se permite seleccionar qué campos se quieren devolver de la tabla y en las consultas esto no se permite ya que se basan en una consulta base (ya se contiene esta información). Lo que se puede hacer es realizar consultas más específicas, como podría ser, en el caso de datos de cliente, realizar dos consultas, una que solamente muestre los clientes que residen en Madrid y otra los que residen en Barcelona. Ambas dependen de una consulta de clientes, pero se especializan en un criterio.

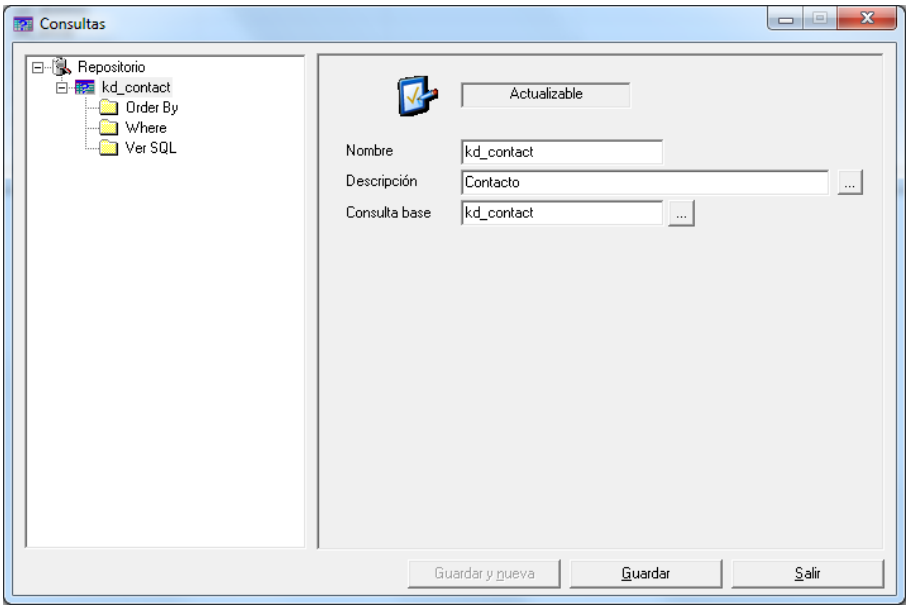


Figura 11: Consulta

4.1.3. Diseño del objeto de negocio, formularios y listados

En este apartado se explica en qué consisten los elementos con los que el usuario realmente va a interactuar y a través de los que se mostrarán y se recolectarán los datos.

- Objeto de negocio:

  El objeto de negocio es un objeto karat que permite gestionar como una entidad única varios niveles de información de la capa de datos.

  Este objeto debe contener toda la lógica de negocio de la entidad que implementa, así como las reglas de validación de ésta. Para cada entidad de la aplicación es necesaria la definición de un objeto de negocio puesto que éste será el que nos permitirá hacer el mantenimiento de los datos asociados a la entidad que representa.

  A continuación se describen los elementos que componen un objeto de negocio:

  - Paneles

  El objeto de negocio está dividido en *paneles*. Como mínimo un objeto de negocio se compone de un panel (recibe el nombre de *HEADER*) y éste puede contener una consulta o ninguna. Es posible que a veces un panel no contenga ninguna consulta, ya sea porque las consultas se tratarán manualmente con código Java a través de las *clases de personalización*[5] o porque los datos que contendrá ese objeto de negocio dependen de una variable de entorno.

[5] Clase de personalización: Se trata de código en Java que contiene funcionalidad añadida (para el objeto de negocio, formulario o listado) que no puede ser controlada mediante las definidoras y que se debe programar aparte.

Una utilidad muy importante que tiene dividir los objetos de negocio por paneles es el filtro por panel. Éste nos permite filtrar los datos que mostrará ese panel si cumplen con una restricción definida, p.ej. que un control tenga un valor concreto, ya sea una constante o el valor que tiene una variable de entorno. Esta funcionalidad en particular se podrá observar durante la fase de desarrollo ya que es usada constantemente en la generación de la aplicación.

- Controles

Como se ha dicho anteriormente cada panel contiene una consulta (por lo general) y para poder trabajar con los datos de cada campo de las tablas a las que apuntan dichas consultas, necesitamos una entidad física que corresponda a cada campo. Dichas entidades son las que llamamos controles. Así pues, cada control está asignado a un campo perteneciente a la consulta.

Como pasaba en el caso de los paneles, es posible que un control no esté asociado a un campo de la consulta. Esto puede ser debido a varios motivos: ya sea que el contenido de ese control sea gestionado mediante una clase de personalización, que el control esté asignado a una variable de entorno o bien que el contenido de este control esté asignado a una *MDQO*[6].

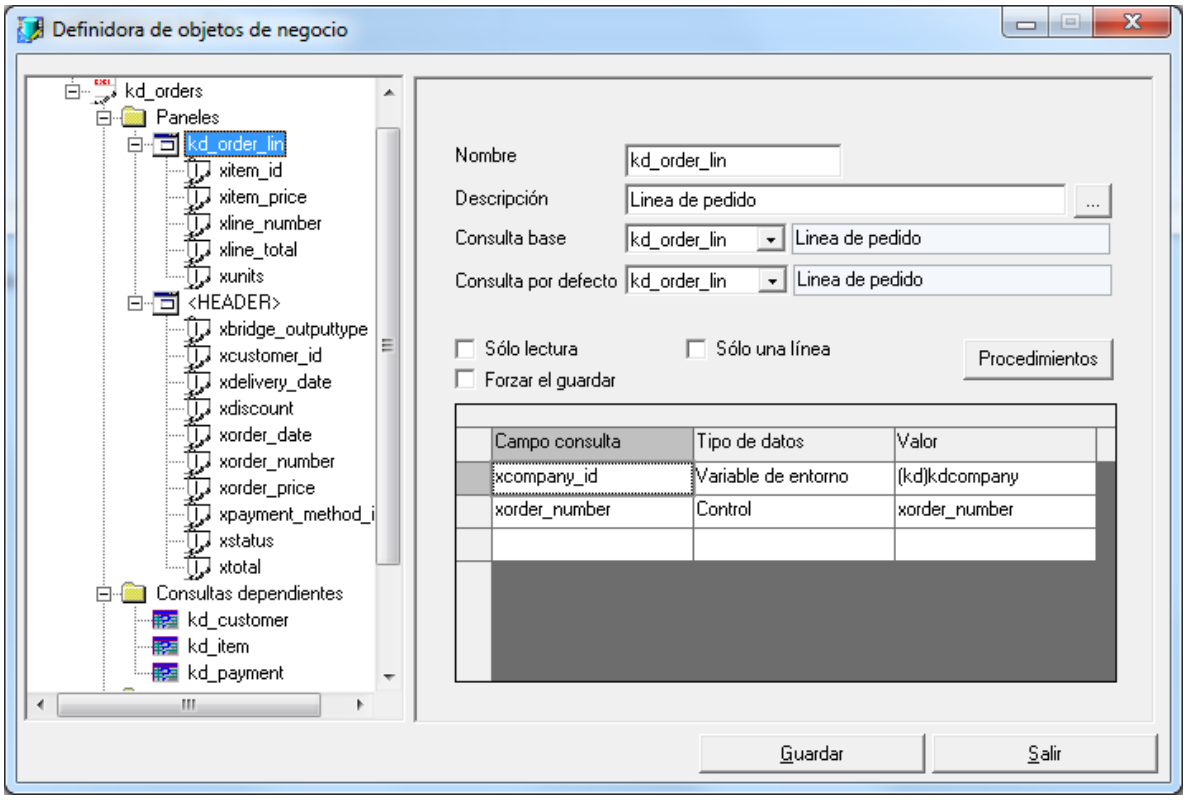


Figura 12: Definidora de Objetos de negocio

[6] MDQO: Se trata de una consulta dependiente. Lo que hacen las consultas dependientes es actualizar el contenido del control a consecuencia de que otro control, determinado en la definición de la consulta dependiente, cambie de valor.

La gestión multiidioma de la aplicación se lleva a cabo en los objetos de negocio, cada control tiene la opción de definir su contenido en ocho idiomas diferentes, en el caso del KSK sólo se definirán los idiomas inglés y castellano.

- Formulario:

Un formulario es la herramienta mediante la que se muestran y se recopilan los datos, y es el encargado de interactuar con las estructuras mencionadas hasta ahora.

Todo formulario depende directamente de un objeto de negocio y su función es mostrar de forma natural y sencilla tanto los datos en sí como la manera en que se puede interactuar con ellos.

Los formularios se pueden dividir en las siguientes partes:

- Datos fijos

En esta sección del formulario se suelen encontrar los datos que son comunes para todo lo que se muestre en el formulario, la información más característica de lo que en él se encuentra.

- Pestañas

Una vez pasada la zona de datos fijos se encuentran las pestañas. Éstas sirven para dividir el formulario en grandes secciones que traten sobre los mismos datos (dependen del mismo objeto de negocio), pero que no tengan un vínculo importante los unos de los otros. Es una manera de crear una cierta independencia física entre los datos mostrados en ese formulario. Cada pestaña sería una gran sección con su funcionalidad particular.

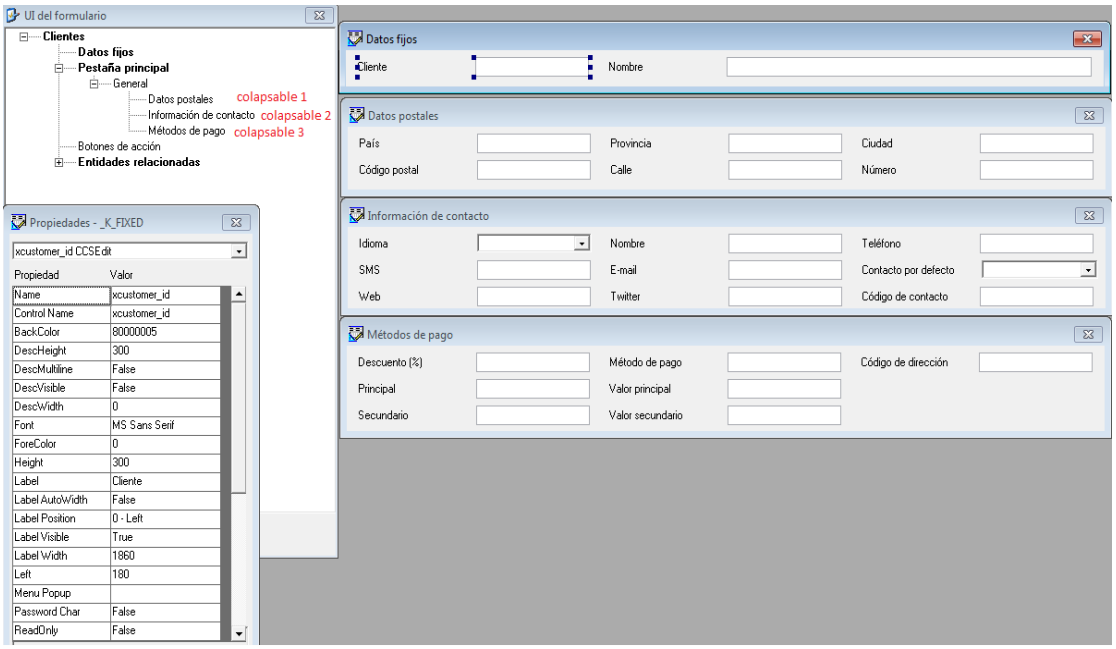


Figura 13: Definidora de formularios

- Colapsables

  Los colapsables son cajas que contienen las casillas donde se escriben o se muestran los datos con los que el usuario trabaja. La característica principal de los colapsables, como el nombre indica, es que se pueden doblar sobre sí mismos y ocultar los datos que contienen.

  Los datos que hay dentro de los colapsables deben cumplir un requisito y es que sean fuertemente dependientes unos de otros, es decir, los colapsables deben contener datos que sean importantes entre ellos. Los datos que no tengan relación directa con los que hay en un colapsable deben colocarse en otro, de manera que si por el motivo que sea, el usuario decide ocultar uno, los datos que restan tengan total sentido entre ellos.

- Listado:

  Un listado *karat* es el objeto que implementa la impresión de un objeto de negocio y permite mostrar su información en distintos formatos.

  Al igual que los formularios dependían directamente de un objeto de negocio, los listados dependen directamente de un formulario.

Figura 14: Definidora de listados

La creación de listados es relativamente sencilla, ya que consiste en dividir la hoja en tantas secciones como se requiera. Cada sección contiene: encabezado de sección, detalle y pie de sección. Según la estructura que requiera cada listado se pueden borrar las secciones o subsecciones que no son necesarias, así como hacer que una sección se repita una vez por página, poner una imagen de fondo a una sección o seleccionar que se desean líneas pautadas para visualizar mejor los datos.

Una vez definida la estructura de la página sólo se debe seleccionar el control que se desea mostrar y arrastrarlo a la posición en la que se desea que se muestre en el impreso y si se quiere mostrar el campo *Valor* o *Descriptivo* en el caso que haya varios

4.2. Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

El principal motivo para escoger Java es que actualmente UNIT4 está migrando todas sus aplicaciones a este lenguaje por su potencia y por su independencia de plataforma.

4.2.1. Eclipse

Todos hemos oído hablar de potentes entornos de desarrollo de software por los que había que pagar costosas licencias como *Visual Studio de Microsoft*. Desde hace un tiempo tenemos la oportunidad de tener una herramienta con potencia similar y de libre distribución. Se trata de eclipse, un Entorno Integrado de Desarrollo (IDE) abierto y extensible, para cualquier cosa y nada en particular. Pese a que eclipse esté escrito en su mayor parte en Java (salvo el núcleo), se ejecuta sobre una máquina virtual de ésta y su uso más popular sea como un IDE para Java, eclipse es neutral y adaptable a cualquier tipo de lenguaje. La característica clave de eclipse es la extensibilidad. IDE eclipse es una gran estructura formada por un núcleo y muchos plug-ins[7] que van conformando la funcionalidad final.

El IDE Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde la funcionalidad está toda incluida, la necesite el usuario o no. Este mecanismo de módulos ofrece una plataforma ligera para componentes de software.

Dado que el departamento de plataforma ha dedicado muchos esfuerzos en crear varios plug-ins para Eclipse y karat, no ha habido ningún tipo de duda a la hora de escoger este entorno de desarrollo para todo lo referente a programar en Java. Para realizar el proyecto se ha escogido la versión “Eclipse IDE for Java EE Developers”, es una versión algo más pesada que la usada habitualmente pero en este caso se requiere el soporte para aplicaciones Web que sólo da esta versión.

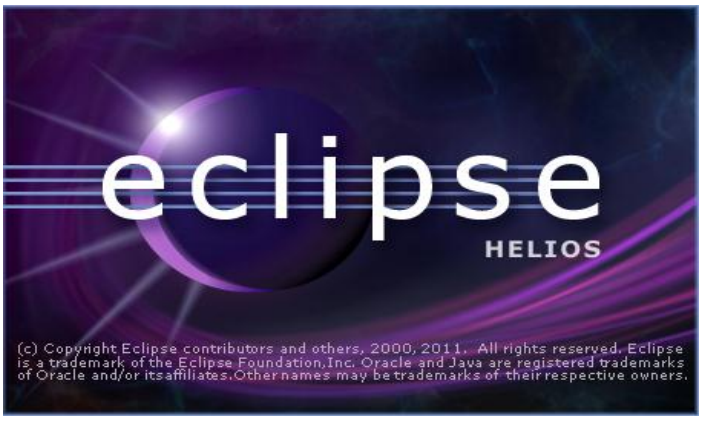


Figura 15: IDE Eclipse Helios

[7] Plug-in: es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

DESARROLLO DE LA APLICACIÓN

Tal y como se ha comentado en la introducción, para la realización del proyecto se ha seguido la metodología Scrum así que se ha dividido el trabajo en Sprints. Dada la naturaleza del proyecto, cada Sprint consta de una parte de diseño, codificación, prueba y documentación.

La estructura que sigue cada Sprint es:

- Tabla con los hitos a realizar
- Cómo se ha realizado
- Fase de prueba
- Documentación (en el caso que sea necesario)
- Muestra del Burndown-chart y comentario del resultado

5.1. Sprint 0

Este Sprint es un Sprint bastante peculiar ya que en éste no se aplica todavía la metodología Scrum debido a que no está implantada del todo en la gestión del proyecto, pero como el resto de tareas sí que se harán con Scrum se ha decidido nombrar a esta fase del proyecto Sprint 0. A consecuencia de ello no hay Burndown-chart para este Sprint de iniciación.

Consideramos objetivos de este Sprint todas las historias a realizar hasta la fecha de inicio del Sprint 1. Dichas historias son las siguientes:

- Diseño del módulo básico de ventas
- Creación de las bases del módulo
- Definición de Objetos de negocio y Formularios
- Creación de Listados

5.1.1. Diseño del módulo básico de ventas

El producto KSK Reloaded representa un modelo básico de compras que consta de varios módulos: empresa, cliente (incluyendo información de contacto, dirección física y de métodos de pago habituales o favoritos de cada uno), productos y pedidos (formados por cabecera y línea de pedido).

Con este producto se intentan recrear las características, necesidades e interacciones entre los datos de las bases de datos que puede tener cualquier cliente que use karat en su negocio y también las herramientas y funciones de que dispone.

A continuación se muestra el modelo de datos del nuevo KSK:

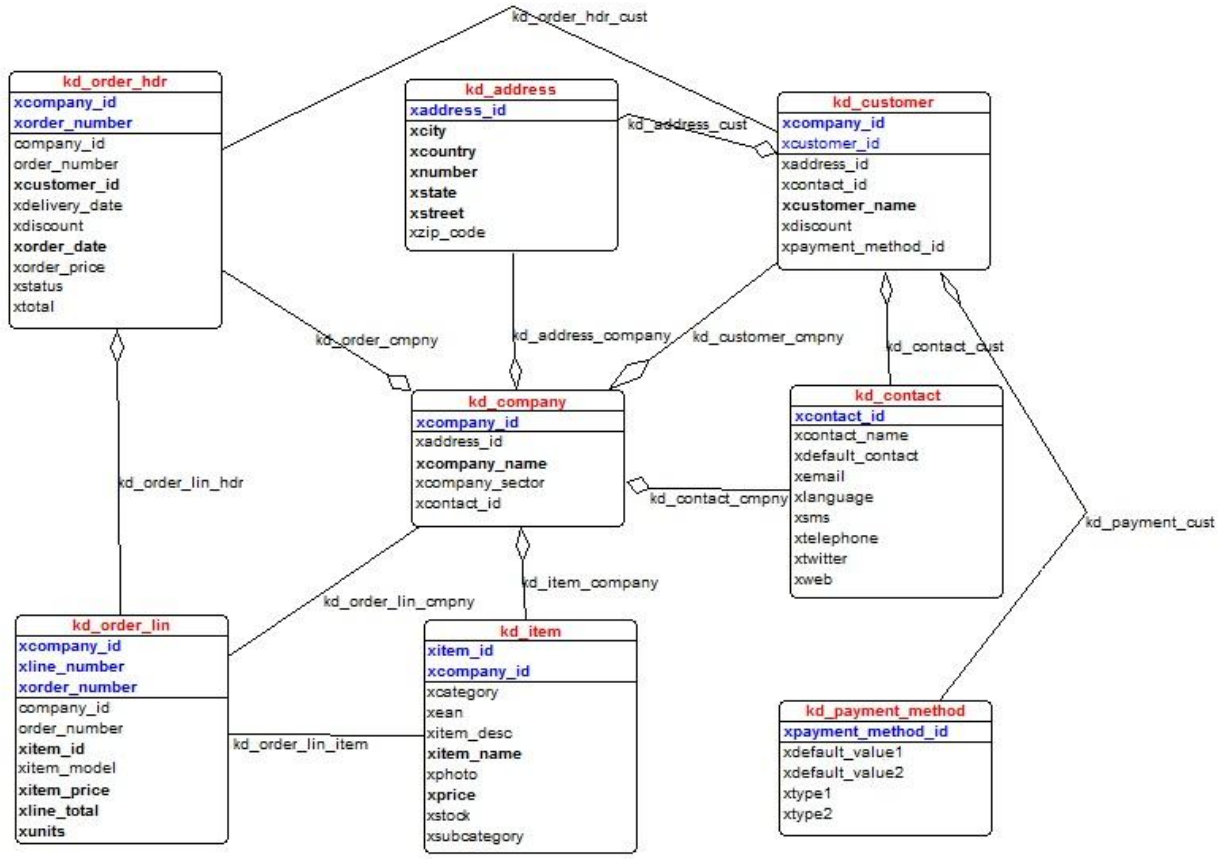


Figura 16: Modelo de datos del Nuevo KSK

Cada módulo del modelo de datos representa una tabla de la base de datos y una entidad con significado por sí misma. Los módulos clientes y empresas están compuestos de su propia información y de una referencia a las tablas “kd_contact” y “kd_address, y en el caso de cliente una referencia extra a la tabla “kd_payment_method”. Los pedidos están compuestos por las tablas “kd_order_lin” y “kd_order_hdr”.

Como se puede observar el nuevo KSK tiene ciertas similitudes con el actual, ya que el nuevo KSK está inspirado en él, añadiéndole ciertas mejoras:

- Se ha añadido información adicional a la empresa. En el producto actual sólo se contempla la identificación y el nombre de ésta, en el nuevo KSK se ha añadido la información de contacto, la dirección física y el sector al que pertenece la empresa.
- Se ha creado una tabla de dirección física aparte. Se ha decidido separar los campos de dirección física de empresas y clientes de sus respectivas tablas para insertarlas en una tabla usada exclusivamente para ello, permitiendo así tener la información más detallada así como hacer búsquedas más concretas.
- Se ha creado una tabla de información de contacto. Como en el caso anterior se ha creado una tabla aparte con la información de contacto permitiendo así más detalle. En esta tabla encontramos datos como idioma, teléfono, e-mail y web que ya existen en el producto actual, y se han añadido nuevos campos:

  - contact_name: Permite saber en cada empresa y/o cliente con que persona se va a contactar, lo que permite un trato más cercano.
  - sms: Permite disponer de un número especialmente para los avisos por mensaje de texto.
  - *twitter*[8]: Dirección de twitter, sistema del que disponen cada vez más empresas que buscan sacar mayor partido de las nuevas tecnologías.
  - default_contact: Permite a cada empresa y/o cliente informar de qué modo prefiere que se pongan en contacto con ellos, lo que proporciona un trato más cercano al permitir adaptarse al cliente.
- Se ha creado una tabla con información del método de pago. Dependiendo del cliente se permite predeterminar hasta dos métodos de pago habituales (Transferencia bancaria, Paypal, Tarjeta de crédito, etc.) y guardar la información de ésta (número habitual de cuenta, nombre de usuario, número de tarjeta, etc.) para poder reembolsar al cliente en caso de algún error sin tener que contactar con él.
- Se han añadido campos adicionales en la tabla de artículos:
  - category y subcategory: Permite clasificar los productos en categorías y subcategorías, lo que da la posibilidad de hacer búsquedas mucho más detalladas por tipología (p.ej. category: Periféricos, subcategory: Impresoras).
  - stock: Seguimiento de la cantidad de producto disponible. Facilita además la puesta en práctica de *Simple Workflow* (a realizar en el Sprint 1), haciendo que se active una tarea de pedir a proveedor al llegar esta cantidad a un número prefijado.
- Se han añadido campos adicionales en la tabla de cabecera de pedido:
  - delivery_date: Permite al cliente solicitar una fecha de entrega posterior a la que sería la de entrega habitual.
  - Status: estado actual del pedido (pendiente de procesar, en curso, finalizado, cancelado) que permite el seguimiento interno del pedido, facilitando saber así en qué estado se encuentra en el caso que haya alguna irregularidad en el proceso completo de entrega al cliente.

5.1.2. Creación de las bases del módulo

En este apartado se detallan los componentes utilizados para la creación del módulo básico de ventas y qué características tiene cada uno para poderlos utilizar posteriormente para la creación de los objetos de negocio y formularios.

Tal y como se ha comentado en el capítulo anterior, para la creación del módulo básico se han utilizado las siguientes herramientas:

- Variables de entorno
- Dominios
- Tablas
- Consultas base y consultas

Para la creación de todas estas estructuras se ha hecho servir una nomenclatura concreta debido a que uno de los objetivos del proyecto es precisamente definir y seguir una nomenclatura uniforme en todos los componentes del KSK.

[8] *Twitter:* es un sitio web de microblogging que permite a sus usuarios enviar y leer micro-entradas de texto de una longitud máxima de 140 caracteres denominados como "tweets".

Para empezar, todos los componentes deben ser nombrados en inglés, ya que el producto se distribuye a nivel global y debe estar en un lenguaje entendible internacionalmente.

Para dar nombre a las variables de entorno se ha usado el prefijo “kd”, siglas de karat demo, antes del nombre de la variable. Para los contadores, dominios y tablas se ha usado el prefijo “kd_” seguido del nombre de lo que se implementa, p.ej. kd_company para el contador, tabla y dominio de empresas.

Para los campos de las tablas no solo se ha utilizado un nombre que identifique claramente qué contiene ese campo, sino que además se ha usado el prefijo “x” delante de cada campo. El motivo por el que se ha decidido usar el prefijo x en cada campo es que hay bases de datos que pueden tener nombres reservados para uso interno, y lo que se busca es que el KSK sea compatible 100% con cualquier plataforma y base de datos, así que se ha tomado esta medida para disminuir las posibilidades de cualquier conflicto con el equipo del cliente.

Variables de entorno

Se ha creado una variable de entorno **kdcompany** cuya función es permitir seleccionar con que empresa se está trabajando en cada momento, sin necesidad de solicitarlo constantemente.

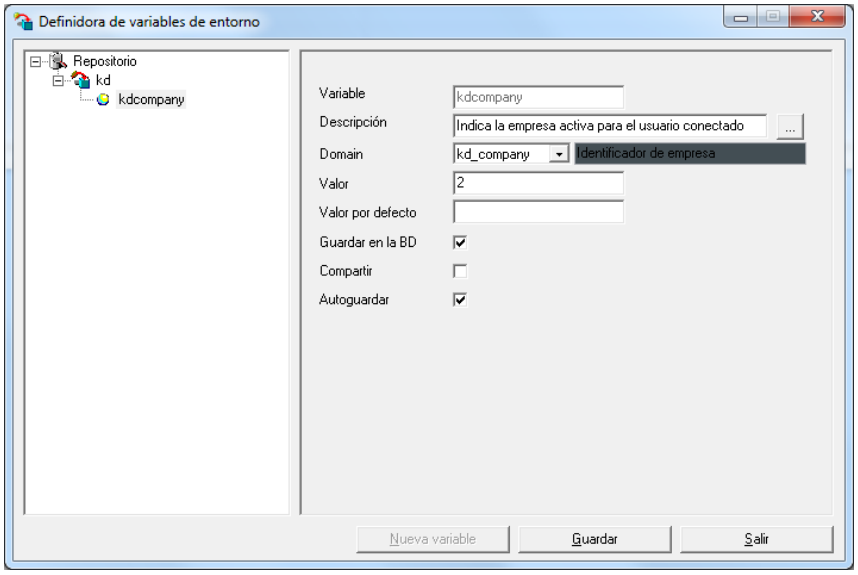


Figura 17: Variable de entorno

Como se muestra en la *Figura 17* se ha configurado la variable de entorno para que se guarde automáticamente en la base de datos cada vez que se le asigne un valor.

Dominios

Para agilizar la creación de las tablas se han definido varios dominios, en concreto uno por cada identificador de cada tabla. En total tenemos los siguientes dominios:

- kdaddress: identificador de dirección física.
- kdarticle: identificador de artículo.

- kdcompany: identificador de empresa.
- kdcontact: identificador de contacto.
- kdcustomer: identificador de cliente.
- kdorder_number: número de pedido.
- kdpayment_method: identificador de método de pago.

La estructura de los dominios es exactamente la misma ya que todos se usan como identificadores numéricos de cada tabla, así que sólo se describe la primera estructura:

kdaddress: identificador de dirección física.

- Tipo de datos: LONG (tipo de dato numérico de mayor longitud que un entero corriente).
- Obligatorio.
- Clase de control tipo: TEXT BOX (caja de texto).
- Máscara de entrada: ###### (admite hasta 6 dígitos).

Tablas

Como se ha comentado en el diseño del modelo de datos, cada módulo de la *figura 16* representa una tabla de la base de datos. A continuación se explica que datos contiene cada una de las tablas:

- kd_company: información propia de la empresa.
- kd_customer: información propia del cliente.
- kd_address: información de dirección física.
- kd_contact: información de contacto.
- kd_article: información de los artículos.
- kd_order_hdr: datos de cabecera de pedido, que contiene la información principal del pedido, sin entrar en los detalles de los artículos que lo componen.
- kd_order_lin: datos de línea de pedido, que contiene los artículos que componen el pedido en sí y qué cantidad de ellos se ha solicitado.
- kd_payment_method: información sobre el método de pago usual, con dos opciones distintas.

Consultas base y consultas

Se ha creado una consulta base por cada tabla cuya finalidad, como se describió en el capítulo anterior, es mostrar todos los campos de cada tabla ordenándolos por su clave primaria.

Así pues tenemos ocho consultas base cuyos nombres son: kdaddress, kdarticle, kdcompany, kdcontact, kdcustomer, kdorder_hdr, kdorder_lin y kdpayment_method.

Con la creación de las consultas base se han creado consultas exactamente con el mismo nombre y la misma estructura. Para el funcionamiento del producto no es necesario definir ninguna más, ya que de la gestión de los campos se encargará cada objeto de negocio.

5.1.3. Definición de Objetos de negocio y Formularios

Una vez definidas las herramientas básicas se puede empezar a diseñar el núcleo de la aplicación, los objetos de negocio, quienes se encargarán de gestionar los datos y los formularios para mostrarlos al usuario. A continuación se describen las estructuras que se han tenido que crear para cada mantenimiento del KSK. El criterio seguido para nombrar los objetos de negocio y los formularios es el habitual prefijo “kd_” seguido de un nombre (el mismo para objeto de negocio y formulario) que los identifique.

- **Selección de empresa activa:** Como se ha comentado en varias ocasiones se ha creado un sistema multiempresa con lo que se pueden gestionar varias empresas a la vez, cada una con sus clientes, artículos y pedidos. Para no tener que informar constantemente con qué empresa se trabaja se ha creado un formulario que fije la empresa con la que se desea trabajar para que la gestión sea más cómoda.

  Objeto de negocio: Primero de todo trataremos el objeto de negocio que trata con la gestión del sistema multiempresa.

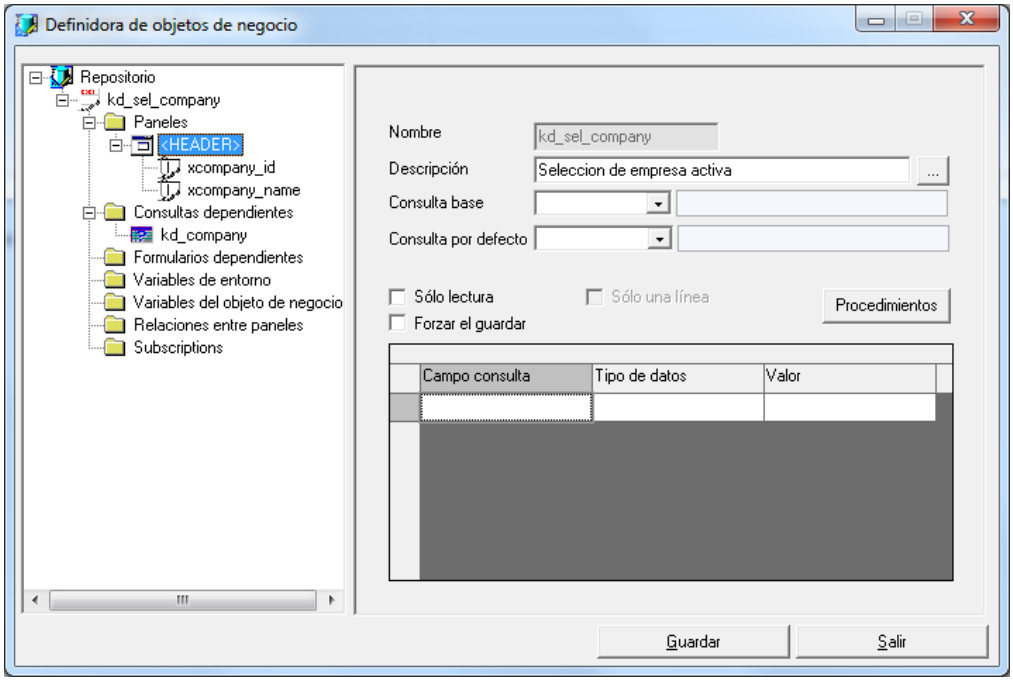


Figura 18: Objeto de negocio de selección de empresa activa

Como se observa en la figura 18 es un objeto de negocio sencillo que consta del panel *HEADER* con sólo 2 controles y ninguna consulta base asociada. Esto es debido a que el campo *identificador de empresa* está asociado a la variable de entorno y se guarda directamente en la base de datos, y a que el nombre se obtiene de una MDQO que muestra el nombre asociado al *identificador de empresa* en la tabla kd_company.

Formulario: Una vez finalizado el objeto de negocio se crea el formulario a partir de éste. Este formulario consta sólo de una pestaña y un colapsable en el que se encuentran los dos controles del objeto de negocio, el identificador y el nombre. Este último como "sólo lectura" ya que se trata de un campo informativo para que el usuario no tenga que memorizar los identificadores. Al pulsar el botón "Aplicar" se graba el valor del identificador en la variable de entorno.

Clase de personalización: Para complementar este formulario se ha añadido una clase de personalización cuyo objetivo es ejecutar una función al hacer clic en el botón "Aplicar" para comprobar mediante una sentencia SQL si la empresa seleccionada existe en la base de datos. En el caso que esta empresa no exista se muestra un mensaje karat (creado mediante la definidora de mensajes karat) informando de que la empresa en cuestión no existe. La clase de personalización se ha nombrado KDFMSel_company (KD de karat demo, FM siglas usadas para nombrar formulario y a continuación el nombre del formulario), el código Java para esta clase de personalización y las que se nombrarán en adelante se puede encontrar en el CD que se adjunta con el proyecto.

A continuación se muestra una imagen del formulario de selección de empresa activa terminado:

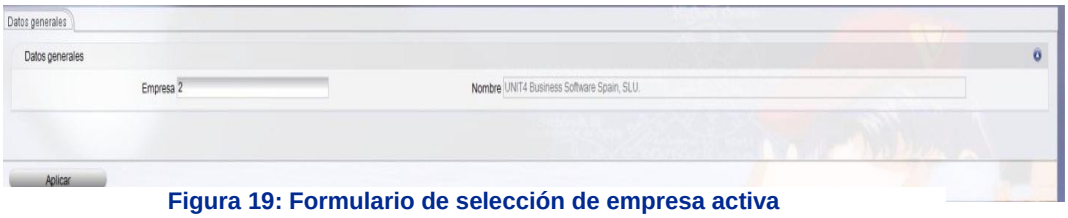


Figura 19: Formulario de selección de empresa activa

- **Mantenimiento de empresas**: Como el nombre indica, este formulario tiene la finalidad de mantener las empresas, es decir, crear, modificar y eliminar las empresas que componen nuestro módulo de ventas.

  Objeto de negocio: Este objeto de negocio está compuesto por tres paneles: el panel principal *Header* que contiene la información de la tabla "kd_company" y con la consulta "kdcompany", el panel *kd_address* que contiene la información de la tabla con el mismo nombre y está enlazado a la consulta "kdaddress" y un último panel *kd_contact* que contiene la información de la tabla con el mismo nombre y consulta "kdcontact". En particular en este objeto de negocio se usa por primera vez un filtro por panel, esto es debido a que las tablas de dirección y contacto contienen todos los datos de dirección y contacto, tanto de clientes como de empresas, así que para mostrar los datos que corresponden a cada uno hay que relacionar el campo *xaddress* del panel Header con el campo con el mismo nombre del panel de direcciones. De esta manera nos aseguramos mostrar los datos que corresponden a cada empresa. A partir de ahora cada vez que en un objeto de negocio se diga que se ha filtrado un panel por un campo, se trata de crear una relación igual a ésta, pero con los datos que correspondan.

  El panel *kd_contact* ofrece una oportunidad excelente para mostrar una funcionalidad añadida que permite vincular los campos e-mail, teléfono y web con las aplicaciones predeterminadas en nuestro ordenador para gestionar dichos servicios.

Otra funcionalidad añadida a este objeto de negocio es el concepto de *Formulario dependiente*, que también se aplicará en los próximos objetos de negocio. Un formulario dependiente no es más que una referencia directa a otro formulario karat que está relacionado con el actual, es decir, en el caso del mantenimiento de empresas, como una empresa tiene clientes, artículos y pedidos, se crea un acceso directo a los mantenimientos de clientes, artículos y pedidos **únicamente** que tengan relación con la empresa actual.

Formulario: este formulario está compuesto por una sección de "datos fijos" que contiene la información común al resto del formulario que son el identificador de empresa y el nombre. Como la gran mayoría de formularios creados, contiene únicamente una pestaña y tantos colapsables como paneles hay en el objeto de negocio, en este caso tres: uno para la información básica de la empresa (sector al que pertenece), otro para los datos postales y el último para la información de contacto.

Por último se han añadido las entidades relacionadas que se han definido en el objeto de negocio.

A continuación se muestra como queda el mantenimiento de empresas:

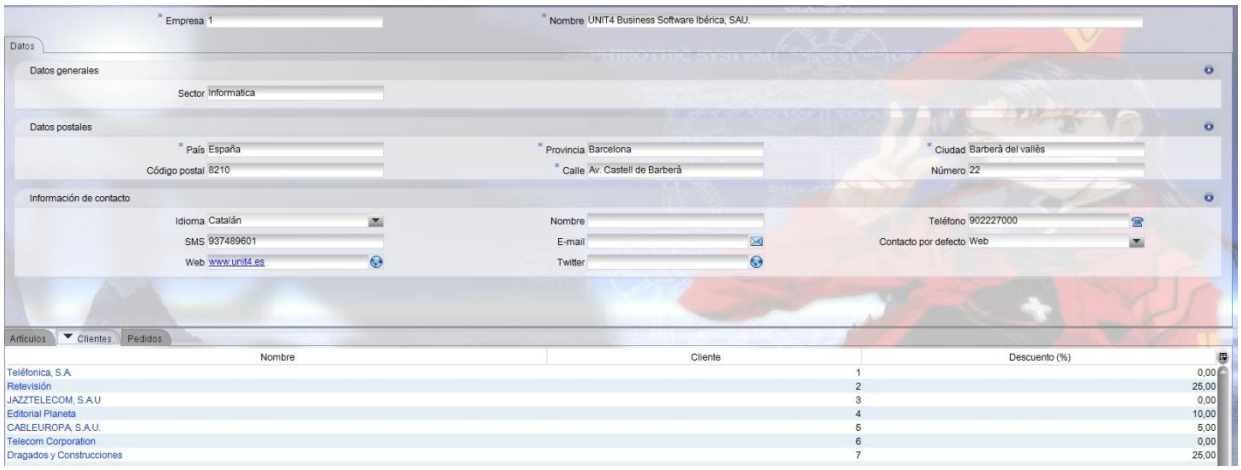


Figura 20: Mantenimiento de empresas

- **Mantenimiento de clientes:** Este mantenimiento tiene una estructura muy similar a la del anterior, tanto a nivel de objeto de negocio como de formulario.

  Objeto de negocio: Este objeto de negocio está compuesto por cuatro paneles, los mismos que el objeto de negocio de empresas pero añadiéndole un panel extra llamado *kd_payment_method* que contiene la información de los métodos de pago del cliente y el descuento (si procede) que se aplica al cliente.

  La estructura de los tres paneles es idéntica al caso anterior, lo único que cambia es que en este caso el panel Header está filtrado por la variable de entorno. Con esto conseguimos que sólo se muestren los clientes pertenecientes a la empresa activa (la aplicación del filtro se puede observar en la figura 21) y la otra diferencia son los formularios dependientes, que en este caso son el mantenimiento de pedidos y empresas (ya que un cliente puede pertenecer a varias empresas, no sólo una).

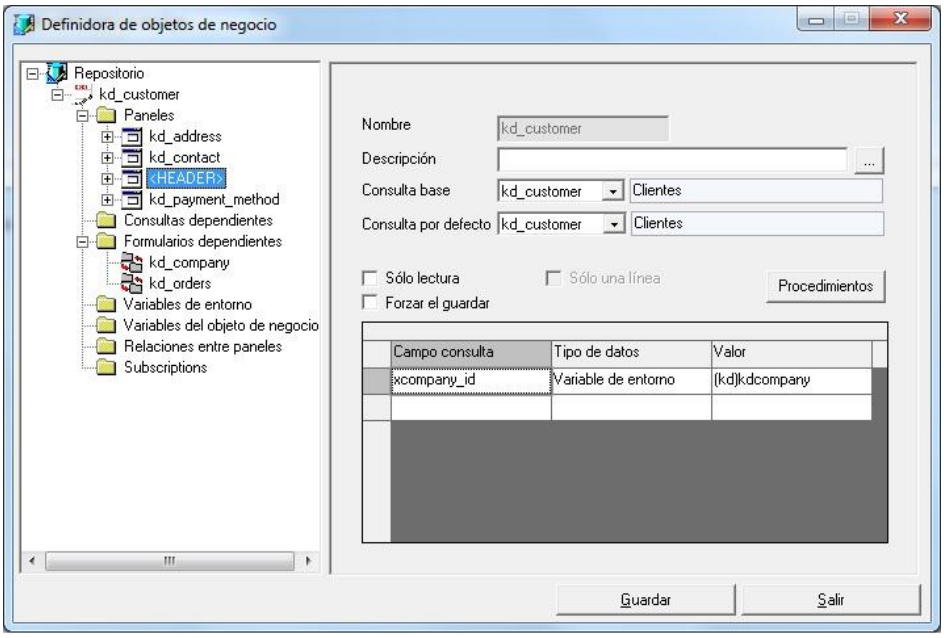


Figura 21: Objeto de negocio de clientes

Formulario: El formulario de clientes está compuesto por los datos fijos de los clientes (identificador y nombre), una pestaña única y tres colapsables, uno por cada panel del objeto de negocio.

- **Mantenimiento de artículos**: Éste se podría decir que es el mantenimiento más sencillo de todos cuya única finalidad es mantener los artículos.

  Objeto de negocio: El objeto de negocio está formado únicamente por el panel Header, obviamente filtrado por la variable de entorno para mostrar sólo los artículos pertenecientes a la empresa activa, y los controles que contienen la información de cada uno de los artículos. Cabe destacar que en este objeto de negocio se encuentra el único control de tipo foto de todo el módulo de ventas.

  Formulario: Al igual que el objeto de negocio, el formulario es muy sencillo, consta únicamente de datos fijos (identificador y nombre) y una pestaña con un colapsable, que no es más que la muestra de todos los controles del objeto de negocio.

  Clase de personalización: La clase de personalización de este mantenimiento actúa sobre el objeto de negocio, así que recibe el nombre de KDBOItem (KD de karat demo, BO de *Business object* y el nombre del objeto sobre el que se encuentra). La función que se ha sobrecargado[9] es *segmentNewRecord* que se ejecuta cada vez que creamos un nuevo artículo, y su funcionalidad es cumplimentar el campo EAN (perteneciente al código de barras) automáticamente. Éste se calcula en función de la empresa a la que pertenece, el número de artículo que es y de una variable aleatoria.

  A continuación se muestra el mantenimiento de artículos:

[9] Sobrecarga: En programación orientada a objetos la sobrecarga se refiere a la posibilidad de tener dos o más funciones con el mismo nombre pero funcionalidad diferente

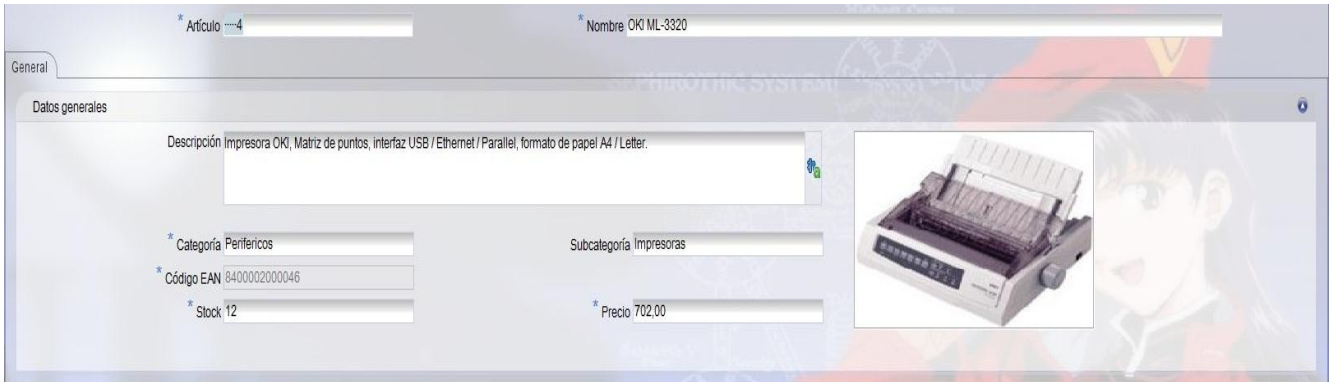


Figura 22: Mantenimiento de artículos

- **Mantenimiento de pedidos:** Este mantenimiento es uno de los más complejos ya que su funcionamiento básico es sencillo, pero es sobre el que más funcionalidades se ha añadido.

  Objeto de negocio: El objeto de negocio de pedidos "kd_orders" está formado por dos paneles, el panel Header que contiene los datos de cabecera de pedido mediante la consulta "kd_order_hdr" y los datos de las líneas de pedido mediante "kd_orders_lin". El panel principal está filtrado, como es costumbre, por la empresa activa, y el panel kd_order_lin está filtrado por la empresa activa y por el número de pedido.

  En este objeto de negocio encontramos de nuevo el concepto de MDQO que en este caso se usa para autocompletar los controles *método de pago* y *descuento* al seleccionar un cliente. Además también se muestra un campo calculado, que no es más que un control que obtiene su valor a través del resultado de una función matemática aplicada sobre otros dos controles. En este caso el total es el resultado de multiplicar el precio unitario del artículo por las unidades solicitadas.

  Formulario: El formulario de pedidos está formado por los datos fijos comunes a todos los formularios, identificador y nombre del cliente, y por dos pestañas: general y *karat Office Bridge*. De esta última se hablará en otro Sprint ya que no forma parte del módulo básico sino de una funcionalidad avanzada.

  En la pestaña general hay tres colapsables. El primero contiene la información de la cabecera de pedido (el panel Header) con la fecha de creación del pedido que se autocompleta con la fecha actual mediante la clase de personalización, la fecha de entrega que se usa en otros Sprint para mostrar funcionalidad avanzada (Simple Workflow) y el estado del pedido que puede encontrarse en: en proceso, procesado, finalizado y cancelado. Este último control también se usa para una funcionalidad avanzada en otros Sprints.

  En el segundo colapsable se encuentra un *Grid*[10] que contiene los datos de las líneas de pedido. Y para finalizar, en el último colapsable, el de importes, se encuentra el importe del pedido, el descuento y el precio final.

[10] Grid: cuadrículas para representar datos en forma de tabla.

Clase de personalización: Para este mantenimiento hay dos clases de personalización ya que hay una para el objeto de negocio KDBOOrders y otra para el formulario KDFMOrders.

- KDBOOrders: En la clase de personalización del objeto de negocio se encuentran varias funcionalidades. Al crear un nuevo pedido se fija la fecha del sistema como fecha de creación, y el estado del pedido se fija en “Pendiente”. A su vez se lleva un control de los datos que componen las líneas de pedido, de manera que el stock se actualiza al grabarlo disminuyendo su stock para un control real de los productos a la vez que calcula el coste final del pedido. Este control se realiza tanto cuando se crea, modifica o elimina una línea de pedido.
- KDFMOrders: En esta clase de personalización se lleva un control del stock similar al del objeto de negocio, centrándose en la primera carga del formulario. Además de este control hay otras funcionalidades pero como no forman parte del módulo básico se describirán en sus correspondientes Sprints.

A continuación se muestra una imagen del mantenimiento de pedidos:

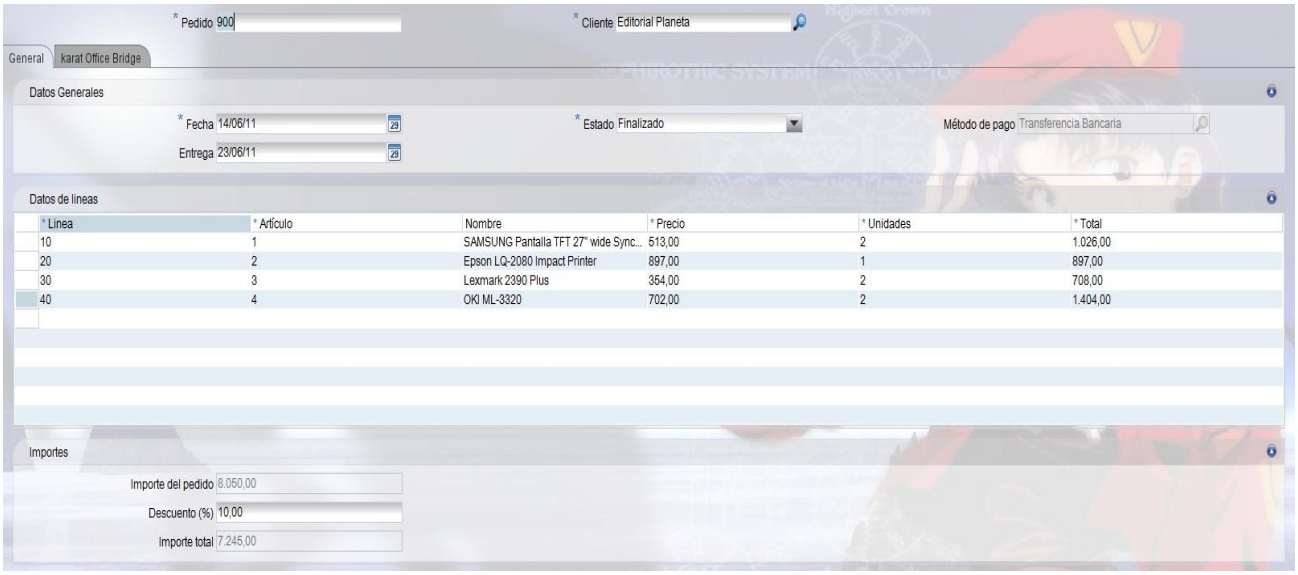


Figura 23: Mantenimiento de pedidos

5.1.4. Creación de listados

Como objetivo final del Sprint 0 se ha creado un listado que muestra los datos de cada mantenimiento, a excepción del de artículos, para el que se han creado tres listados distintos.

Como se ha explicado en el capítulo anterior, la creación de listados es bastante sencilla e intuitiva, consiste en crear un encabezado de página con el título del listado en la esquina superior izquierda, y luego una sección con un encabezado donde se informa de lo que se va a mostrar en esa columna y un previo detalle donde se muestra el valor a mostrar. Para finalizar, en la esquina inferior derecha, se informa del número de página y de la fecha.

Para darle un aspecto más corporativo se han vinculado los listados a una plantilla que tiene el logotipo de la empresa y un adorno al pie de página para darle un toque más elegante.

A continuación se mostrarán dos imágenes de un mismo listado, primero en su vista en la definidora y luego su resultado final una vez procesado. En el CD del proyecto se adjuntan también unos ".pdf" con los listados completos.

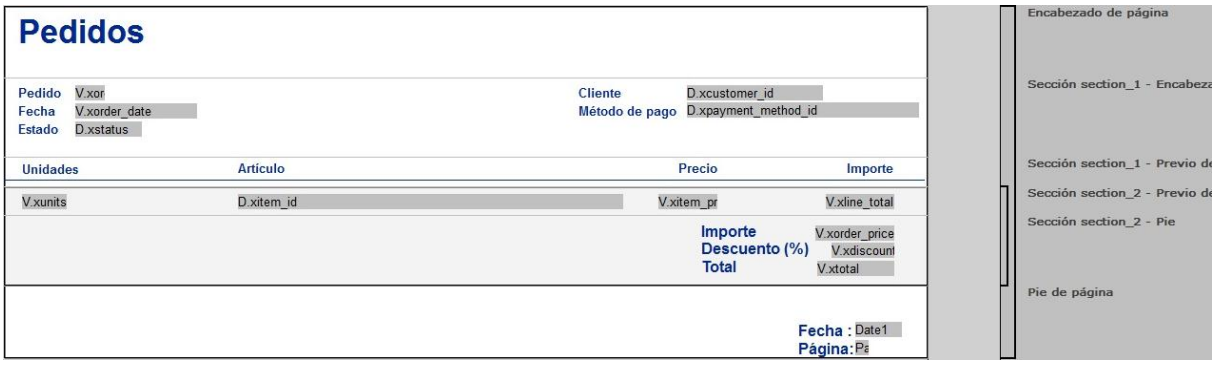


Figura 24: Definidora de listado de pedidos

Figura 25: Listado de pedidos

5.1.5. Fase de prueba

La fase de prueba para este Sprint ha consistido en probar cada mantenimiento y comprobar si los resultados mostrados eran los esperados, así como tratar de introducir valores incorrectos en algunos campos y comprobar que el tipo de datos de cada uno está protegido mediante máscaras.

El resultado de las pruebas ha sido satisfactorio y la aplicación funciona correctamente, excepto la actualización de stock que a veces no muestra el resultado esperado, se anota el error y se solucionará en Sprints posteriores.

5.1.6. Documentación

Para este Sprint se han generado dos documentos que se incluyen en el CD del proyecto, documento de requisitos y documento de análisis, nombrados 188841_KSKReloadedRv1.0 y 188841_KSKReloadedAv1.0 respectivamente.

5.1.7. Burndown chart

Para este Sprint como se ha comentado al principio, no hay Burndown chart ya que se trata de un Sprint simbólico y aun no se ha implantado la metodología Scrum.

5.2. Sprint 1

Este es el primer Sprint real del proyecto así que de aquí en adelante se sigue la estructura nombrada al principio de la fase de desarrollo

5.2.1. Tareas a realizar

A continuación se muestra una tabla con las historias a realizar en este Sprint y el tiempo invertido en puntos[11]:

Historia	Descripción	Coste
H233	Un listado de artículos con código de barras.	0,25
H234	Consolidar los listados y los formularios de KSK Reloaded.	0,50
H235	Incorporar el uso de un MO.	0,25
H236	Formación SW y ver el uso actual en el KSK.	0,50
H237	Incorporar SW. Una regla de negocio para validar un pedido. En caso de exceder un cierto importe generar un aviso al responsable.	0,25
H238	Incorporar SW. Regla de tiempo para la verificación diaria del stock.	0,25
H239	Incorporar SW. Diálogo de validación de stock con llamada a programa y visualización del estado.	0,50
H240	BUGS Sprint 1	0,25

[11] Punto: unidad de tiempo utilizada en la metodología Scrum. Un punto equivale al trabajo realizado por dos personas en una jornada laboral, es decir, 16h de trabajo.

H241	Reuniones periódicas Sprint 1	0
Total		2,75

Tabla 11: Historias del Sprint 1

5.2.2. Fase de codificación

- **Listado de artículos con código de barras**

  El objetivo de esta historia es realizar un listado que genere una etiqueta por cada artículo con el nombre, precio y el código de barras, y aprovechar al máximo el papel. Dadas las dimensiones de cada etiqueta, 7cm de acho, y del área segura de impresión, 2cm por cada lado, se ha configurado el listado para que imprima en dos columnas.

  Se ha definido únicamente una sección de previo detalle, debido a que este listado no requiere de cabecera ni de pie. Al igual que en la creación de los listados del capítulo anterior los campos de nombre y precio se han creado arrastrando el objeto de la ventana de controles a la vista previa del listado, y lo mismo se ha realizado con el campo *xean* (el que contiene el código de barras). La única diferencia es que para que el código de barras se muestre como tal, se ha usado una propiedad que incorpora la definidora de listados karat, que es marcar el control *xean* como un control tipo "código de barras". De esta manera el mismo listado se encarga de la conversión del contenido del control al código de barras que se acostumbra a ver en las etiquetas de los productos.

  A continuación se muestra el aspecto del listado de artículos con código de barras:

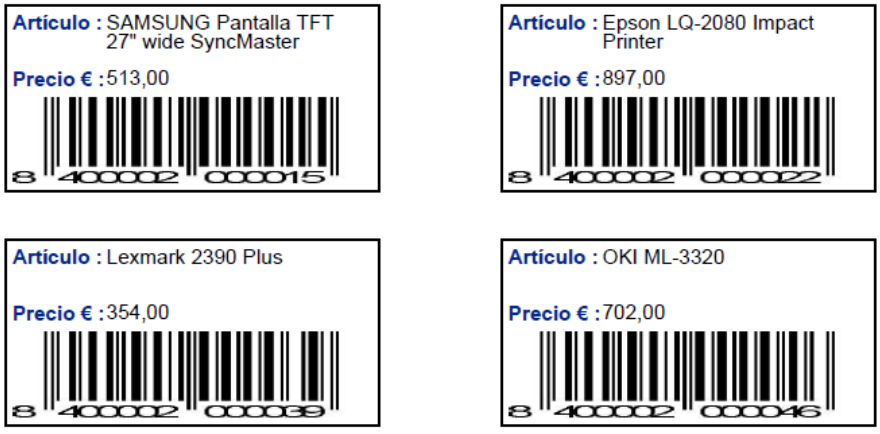


Figura 26: Listado de artículos con código de barras

- **Consolidar listados y formularios de KSK Reloaded**

  El objetivo principal de esta historia es corregir los errores encontrados en la fase de prueba del Sprint 0 y revisar la clase de personalización del mantenimiento de pedidos ya que se detectaron casos en los que no se actualizaba correctamente el stock.

- **Incorporar el uso de un Modelo Organizacional**

  Una de las funciones que se desea mostrar en el nuevo KSK es el modelo organizacional que no consta en el KSK actual. Como resultado se ha creado un modelo inspirado en cualquier empresa del mercado. A continuación se muestra un diagrama del modelo Organizacional:

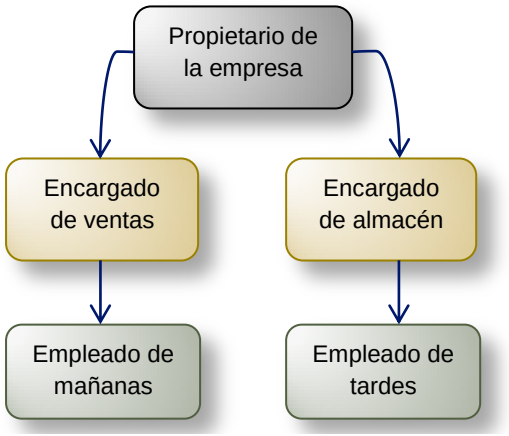


Figura 27: Diagrama del modelo organizacional

  En este modelo hay dos personas con el rol encargado, cada uno cumple un cometido distinto en la empresa y recibirá los avisos necesarios para la gestión de sus departamentos, dos personas más con el rol de empleado y el propietario de la empresa, que tiene el rol de encargado pero recibirá las alertas de todos los departamentos.

  Para la creación de los roles se ha usado la definidora de modelo organizacional. Primero hay que definir cada rol con su nombre, y a continuación los usuarios, con sus nombres, apellidos, contraseñas y nombre de inicio de sesión. Una vez definidos los usuarios hay que indicar a que rol pertenece cada uno.

- **Una regla de negocio para validar un pedido**

  Simple Workflow es una tecnología karat que permite ejecutar un programa determinado cuando se cumple una condición sobre un objeto de negocio. En el caso de esta historia, se pretende ejecutar un programa que envíe un mensaje karat al encargado de ventas cuando un pedido excede un importe determinado y otro al propietario de la empresa.

  Para definir la regla de negocio se sigue una definidora muy intuitiva y se definen las condiciones que se desean. El primer paso es darle un nombre a la regla de negocio. Siguiendo la nomenclatura marcada por Walnut se ha asignado kdbr_checkorder (“br” proviene de business rule), el nombre del objeto de negocio sobre el que se aplica para que cargue los controles y si está activada o no, ya que se puede definir y dejarla desactivada por los motivos que sea.

  En condiciones se define cual es la condición para que se active la regla de negocio, en este caso que el pedido sea superior a 3000€ y para finalizar, en acciones, los parámetros que necesita el programa para funcionar.

  Clase de personalización: El programa KDPCheck_order (p de program) está definido totalmente en Java y se encuentra en el CD adjunto.

El programa envía un mensaje predefinido (en español e inglés) a todos los usuarios que forman parte del rol de encargados.

A continuación se muestra una imagen de la definidora:

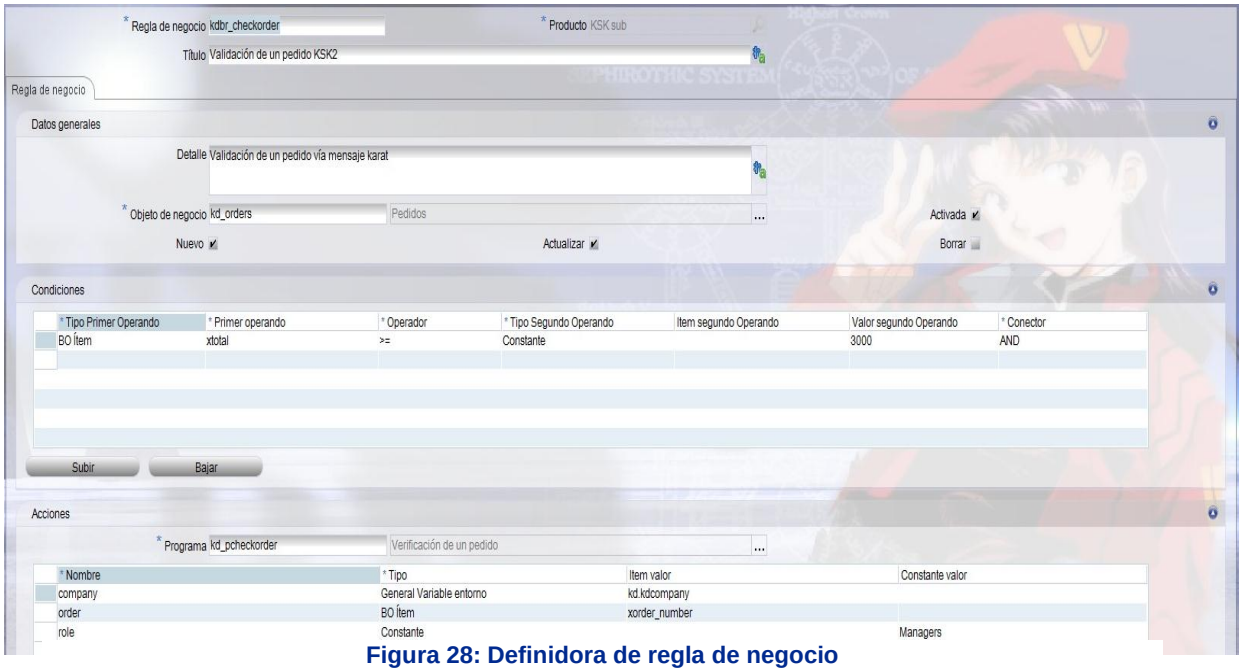


Figura 28: Definidora de regla de negocio

- **Regla de tiempo para la verificación diaria del stock**

El objetivo de esta historia es muy similar a la anterior, la única diferencia es que en este caso el objetivo de la tarea es comprobar el stock de todos los artículos y enviar un mensaje karat con el listado de éstos. La regla de negocio se define igual, la diferencia es que en este caso se usa la definidora de regla de tiempo que nos da la opción extra de fijar con qué frecuencia se debe ejecutar el programa.

Clase de personalización: en este caso se ha nombrado “KDPCheck_stock”, el programa consulta mediante una sentencia SQL el stock de todos los artículos de la empresa enviada como parámetro y envía un mensaje al propietario de la empresa y al encargado del almacén con el listado de los artículos con stock inferior a la cantidad definida.

A continuación se muestra el mensaje enviado:

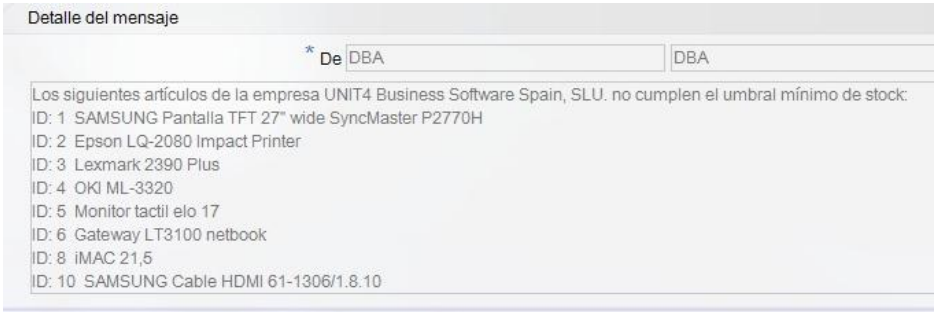


Figura 29: Mensaje enviado tras la ejecución de la regla de tiempo

- **Diálogo de validación de stock con llamada a programa y visualización del estado**

  En esta historia se persigue el mismo objetivo que en la anterior con la diferencia que se desea ejecutar el programa manualmente y ofreciendo la posibilidad de pausar, reanudar o finalizar la ejecución. Para conseguir la gestión manual del proceso se ha requerido definir un nuevo objeto de negocio, crear un formulario con él y para finalizar asignar la clase de personalización al formulario.

  Objeto de negocio: Se ha creado un objeto de negocio “kd_check_stock” sin consulta asociada ya que no se requiere consultar la base de datos, el único dato que se requiere es la empresa activa y para ello usamos la variable de entorno.

  El objeto de negocio contiene el control “min_stock” en el que se introduce la cantidad mínima de stock a controlar y el control “sleep” que es un retraso entre pasos de la ejecución para poder visualizarla mejor.

  Formulario: El formulario consta de una sola pestaña y un colapsable que contiene los dos controles “min_stock” y “sleep” y un campo de texto de solo lectura en el que se imprime el estado actual de la ejecución.

  Para permitir interactuar con la ejecución del programa se han añadido cuatro botones al formulario: Iniciar, Pausar, Reanudar y Eliminar.

  Clase de personalización: La clase de personalización del formulario únicamente incluye una función que comprueba qué botón se ha pulsado para realizar la función correspondiente. Para evitar pulsar botones indebidos se inhabilitan los botones que no tienen sentido según cada situación en la que se encuentra el programa.

  Además de la clase de personalización se incluye una clase Java “KDPCheck_stock” que es la que realiza la comprobación del stock. Las bases de esta clase son los mismos que la anterior, con alguna modificación que añade un control paso a paso de la ejecución y su monitorización en la ventana principal.

  A continuación se muestra el formulario en ejecución:

Figura 30: Verificación de stock en ejecución

5.2.3. Fase de prueba

La fase de prueba para este sprint ha consistido en pruebas de esfuerzo de los formularios y comprobar que los resultados sean los deseados. Se han generado varios ficheros de prueba con los pasos que se han seguido y el resultado de las ejecuciones.

5.2.4. Documentación

Los archivos generados durante la fase de prueba se encuentran en el CD del proyecto y se han nombrado: H0234_ConsolidarListados, H0237_ValidarPedido_VU, H0238_Reglas TiempoStockyFechas, H0239_ComprobarStock_VU.

5.2.5. Burndown chart

A continuación se muestra el Burndown-chart de este sprint y se comenta el resultado.

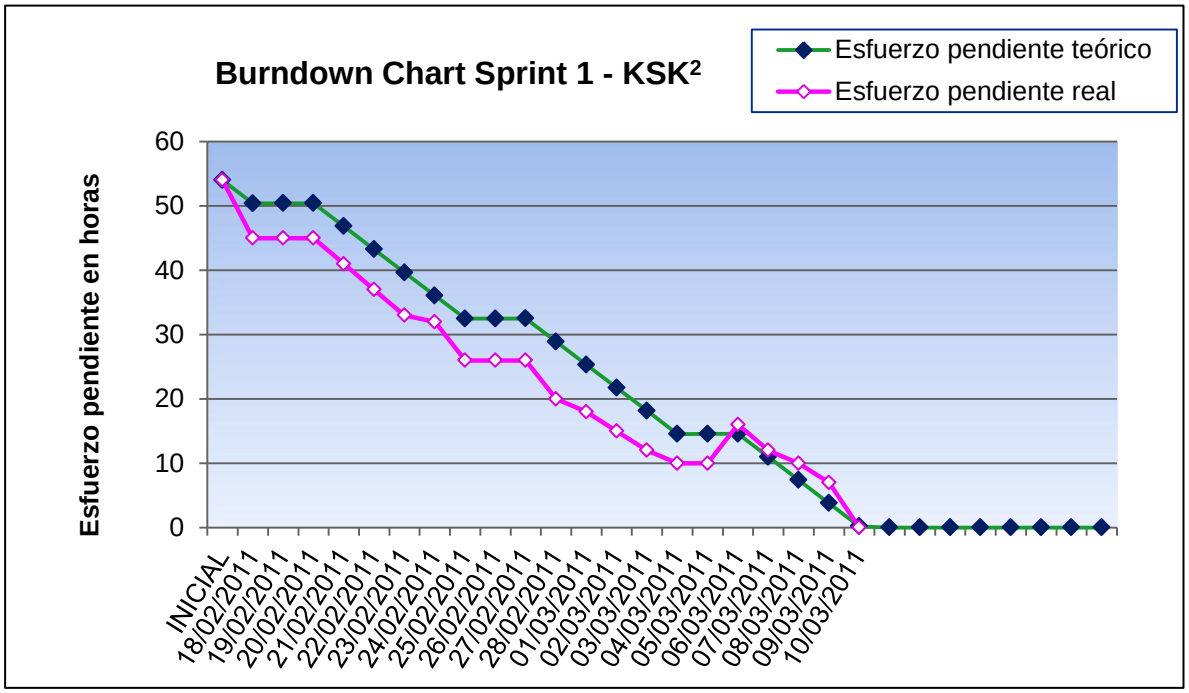


Figura 31: Burndown-chart Sprint 1

Como se observa en el gráfico el trabajo realizado ha ido siempre por delante del teórico. El 6 de marzo se observa un pico de subida, esto se debe a que como se han terminado las historias del Sprint se ha añadido una nueva que ha consistido en realizar un volcado de datos de la antigua base de datos a la actual para hacer las pruebas con nombres reales tanto de productos como de empresas.

5.3. Sprint 2

5.3.1. Tareas a realizar

A continuación se muestra la tabla con las historias a realizar en este Sprint y el tiempo invertido en puntos.

Historia	Descripción	Coste
H244	Uso de atributos de BO en KSK	0,25
H249	Formación Escritorios y Cajas y ver el uso actual en el KSK.	0,25
H245	Escritorio de pedidos por cliente	1,00
H250	Formación WS y ver el uso actual en el KSK.	0,50
H251	Incorporar uso de WS	1,00
H252	BUGS Sprint 2	0,25
H253	Reuniones periódicas Sprint 2	0
	Total	3,25

Tabla 12: Historias del Sprint 2

Varias historias de este Sprint tienen que ver con los *escritorios* karat. Un escritorio es una especie de interfaz que permite visualizar la gran mayoría de los datos sin tener que navegar por cada formulario. Una característica importante de los escritorios es su diseño modular. Están formados por cajas donde cada una contiene la visualización de un formulario. Estas cajas se relacionan unas con otras, de manera que cuando se selecciona un dato en una, el contenido de las cajas relacionadas con ésta se actualiza para mostrar la información sobre los datos que estén relacionados con la selección. Como se ha dicho los escritorios sirven para visualizar, es decir, no se puede crear, modificar ni eliminar datos en estos, para ello hay que acceder al formulario correspondiente. Otra característica muy importante de los escritorios es que las cajas se pueden mover por el escritorio y colocarlas donde se desee arrastrándolas. El sistema que se utiliza para la gestión de las cajas se llama Boxapp.

A continuación se muestra el sistema Boxapp de los escritorios:

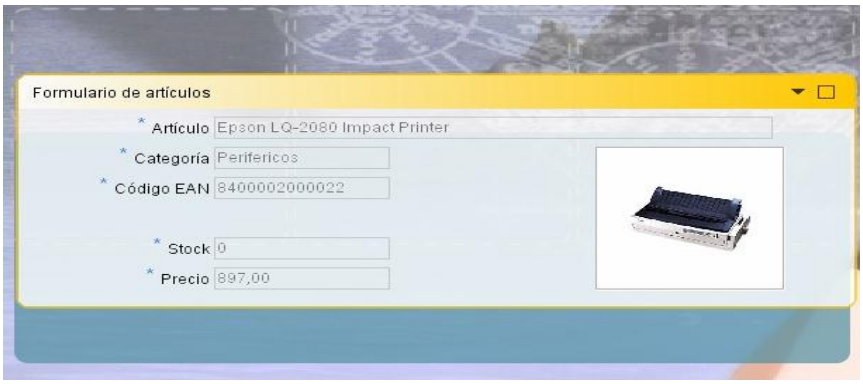


Figura 32: Sistema Boxapp

5.3.2. Fase de codificación

- **Uso de atributos del Objeto de negocio en KSK**

  Como se ha comentado en el punto anterior, una característica de las cajas que componen los escritorios son las relaciones entre ellas y cómo se actualizan los datos entre las cajas seleccionadas. Para ello hay que definir unos parámetros de entrada y salida en cada caja. Los parámetros de entrada definen qué valores provenientes de otras cajas harán que el contenido de la caja actual se actualice, y los parámetros de salida definen qué valores de la caja que estamos configurando queremos que sirvan como entrada para otras cajas.

  Como es lógico, algo imprescindible llegado este punto, es que si se relaciona un parámetro de entrada con uno de salida, éstos deben de ser del mismo tipo, sino la aplicación no funcionará de una forma lógica. Para forzar que los datos de entrada y los de salida sean los mismos, existe el concepto de atributo, que no es más que forzar a un parámetro a tener un tipo de datos concreto.

  Así pues, en esta historia se han definido como atributos todos los parámetros de entrada y salida de cada formulario. Los datos que se han definido como atributos son los identificadores (de cliente y artículo) y el número de pedido.

- **Escritorio de pedidos por cliente**

  El objetivo de este Sprint es la creación de un escritorio que muestre la estructura del modelo de ventas que se ha creado en los Sprints anteriores. Para ello se ha usado la definidora de Boxapp con la que se indica el nombre del formulario sobre el que se hace la caja, y la definición de los parámetros de entrada y salida, usando los atributos creados anteriormente.

  Una vez definidos los parámetros de entrada y salida se han insertado las cajas en el escritorio y se ha procedido a relacionar los atributos de entrada de una caja con los de otra para que la interacción entre ellas funcione correctamente.

  A continuación se muestra un ejemplo de cómo se relacionan los parámetros:

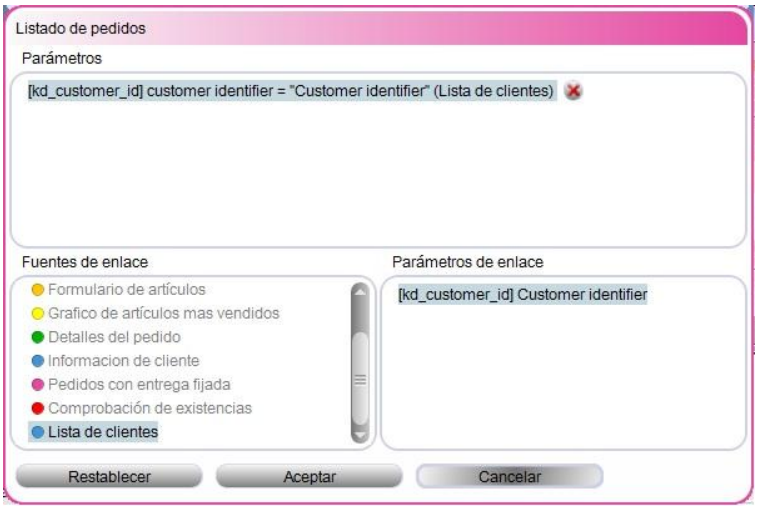


Figura 33: Enlaces entre cajas del escritorio

En la parte superior se puede observar cómo se realiza la asignación. A la izquierda de la igualdad se encuentran los parámetros de entrada y a la derecha en un principio no hay nada hasta que seleccionamos un atributo de la pestaña “Fuentes de enlace”, una vez señalada una fuente se asigna a la derecha de la igualdad y la relación entre las cajas queda definida.

Por último cabe decir que las cajas son unas vistas reducidas de los formularios originales, así que ha sido necesario crear una vista nueva por cada formulario del que se ha creado una caja. No se va a hacer énfasis en la creación de dichas vistas ya que es igual que la creación de los formularios anteriores sólo que seleccionando más estrictamente qué campos son necesarios para dar una visión reducida, pero completa, del formulario en sí.

A continuación se muestra el escritorio del trabajador:

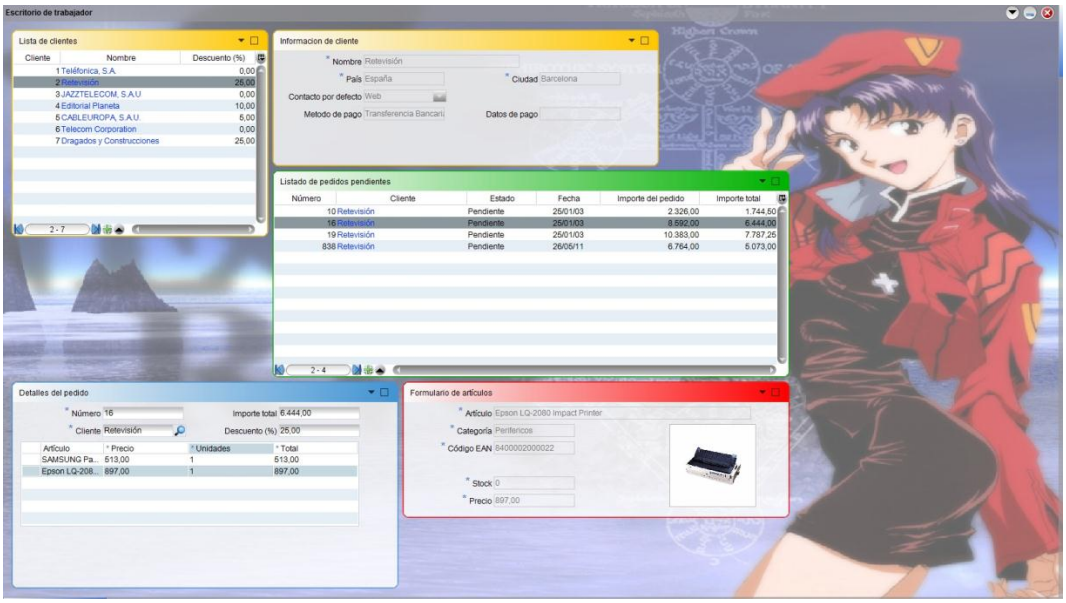


Figura 34: Escritorio del trabajador

Además de crear un escritorio para el trabajador se ha optado por hacer un escritorio más, con funciones especiales, que corresponderá a los usuarios con rol encargado. Este escritorio es similar al anterior con algunas cajas más.

Para empezar se ha creado una caja nueva de listados de pedido en la que se muestra únicamente los pedidos con fecha de entrega fijada en una fecha posterior a la del escritorio. Esto nos permite mostrar que no sólo se pueden usar atributos como datos de entrada sino que también es posible usar datos del sistema, como en este caso la fecha. Además se ha creado una vista reducida del formulario que permite comprobar el stock, sólo que en este caso, al ser una vista reducida, se ha decidido eliminar la opción de pausar, reanudar y eliminar la ejecución del programa.

Por último se ha creado una caja con un gráfico de las ventas. Esta caja sirve para mostrar también que se pueden introducir datos que no interaccionan con otros formularios ya que el gráfico se gestiona mediante código HTML y una pequeña sentencia SQL que comprueba los pedidos uno a uno.

A continuación se muestra el escritorio de encargado:

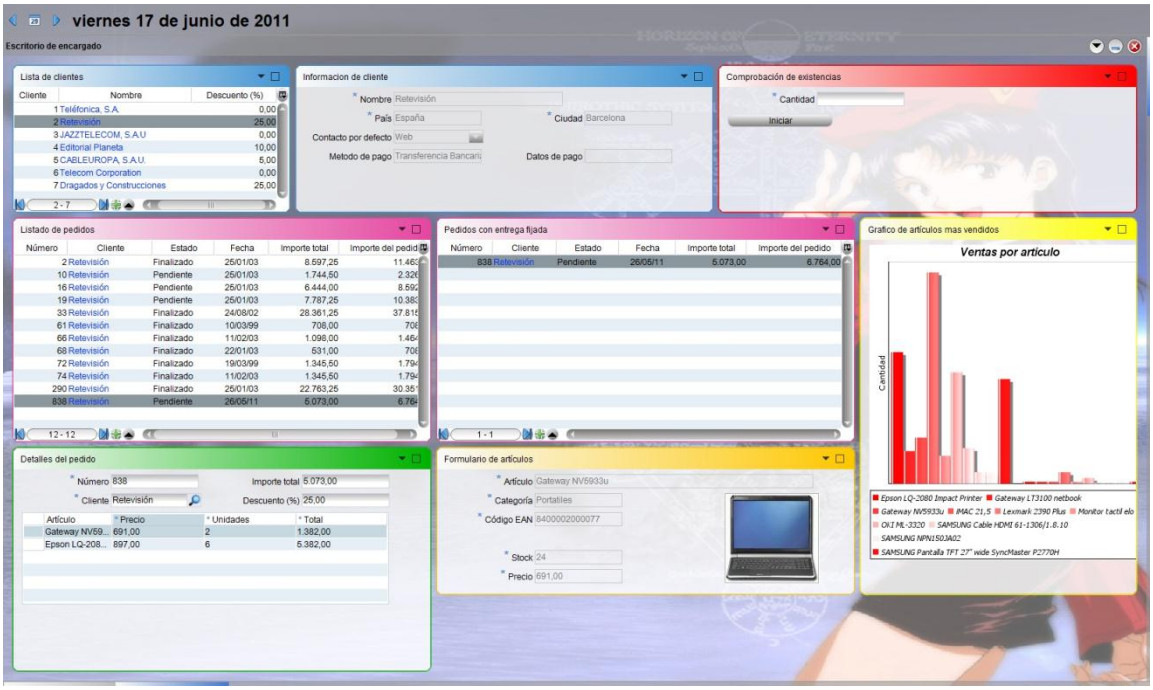


Figura 35: Escritorio del encargado

- **Incorporar uso de Web Services**

En esta historia se pretende mostrar cómo se pueden cargar datos sin necesidad de usar las consultas para acceder a la base de datos. Para ello se ha decidido crear un formulario de artículos idéntico al que ya se ha creado, pero sin consulta base asociada al objeto de negocio, de modo que para obtener los datos se utilizaran los Web Services. Esto es posible gracias a los servicios web, que consisten en crear una página en formato XML con todos los datos de los artículos para su posterior carga en función del artículo que se desee.

Cabe destacar que la complejidad de esta historia ha resultado en la imposibilidad de ser terminada en el tiempo que se estimaba para su creación, ya que se ha utilizado todo en comprender el concepto de Web Service y en su funcionamiento.

5.3.3 Fase de prueba

La fase de prueba de este sprint ha consistido en realizar las verificaciones para cada caja de cada escritorio, que incluyen comprobar que se muestren los datos correctamente, que el tamaño de las cajas se ajuste a las vistas reducidas y ampliadas y que los enlaces entre cajas funcionen correctamente.

5.3.4. Documentación

Fruto de la fase de prueba se ha generado un documento de verificación unitaria, que se encuentra en el CD del proyecto, con el nombre H0259_Escritorios_VU.docx

5.3.4. Burndown chart

A continuación se muestra el Burndown chart del Sprint y se comenta el resultado:

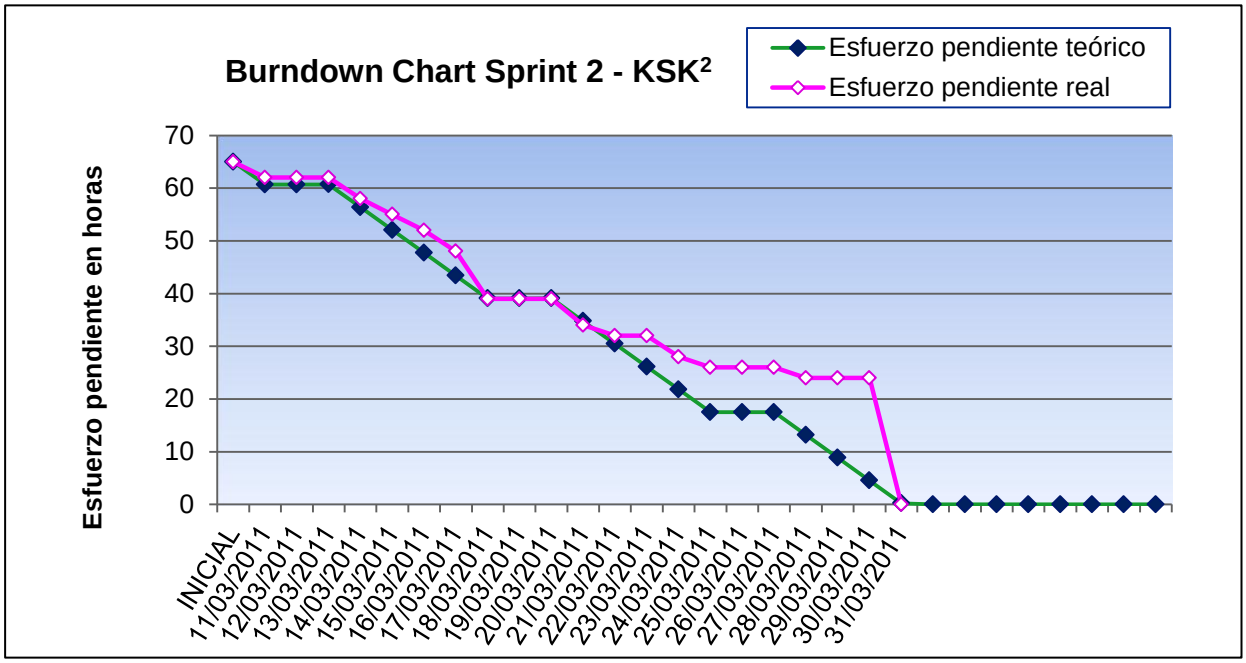


Figura 36: Burndown chart Sprint 2

Como se ha comentado en la última historia, el proceso de comprensión y creación del Web Service ha sido más complejo de lo que se esperaba, lo que ha resultado en una mayor inversión de horas en la formación sobre la materia que se refleja al final del gráfico. Este hecho nos permite comprobar que en Scrum todo queda reflejado y cómo se actúa cuando una situación como ésta sucede.

5.4. Sprint 3

5.4.1. Tareas a realizar

A continuación se muestra la tabla con las historias a realizar en este Sprint y el tiempo invertido en puntos.

Historia	Descripción	Coste
H264	Completar formulario que vía WS muestre los datos de un artículo.	0,25
H258	Formación acerca de cuadro de mando y ver el uso actual en el KSK.	0,25
H266	Añadir cuadro de mando, uso de servicio, Widgets y Workspaces.	1,00
H263	Incorporar ejemplo sencillo de uso de KOB.	0,50
H265	Incorporar búsqueda rápida.	0,25
H267	Despliegue del nuevo producto y realizar copia de seguridad de las clases de personalización	0,25
H268	Reuniones periódicas Sprint 3	0
H270	BUGS Sprint 2	0
	Total	2,50

Tabla 13: Historias del Sprint 3

Este Sprint destaca por varios motivos. Para empezar se puede observar cómo la primera historia trata de los Web Services, que es la historia que se quedó a medias en el Sprint anterior y como se explicaba en los conceptos de Scrum, cuando una historia no se acaba pasa a ser la primera en el siguiente Sprint.

Otro detalle por el que destaca este Sprint es porque contiene menos puntos que los anteriores, esto es debido a que la duración sigue siendo de tres semanas, igual que el resto, pero durante el transcurso de este Sprint se encuentra Semana santa, lo que hace que se puedan dedicar menos horas.

En este Sprint aparte de terminar los Web Services, se habla de *Widgets* y de karat Office Bridge. Los Widgets, como ya se explicó anteriormente, son marcadores dinámicos que permiten monitorizar ciertos datos de los objetos de negocio. karat Office Bridge se trata de un motor de integración, diseñado por karat, para los documentos ofimáticos que permite exportar los datos de un formulario directamente a un documento de texto o hoja de cálculo (ya sea de Microsoft Office u Open Office).

5.4.2. Fase de codificación

- **Completar formulario que vía Web Services muestre los datos de un artículo**

  Para esta historia se parte del conocimiento adquirido en el Sprint anterior.

Objeto de negocio: Como todo formulario karat ha sido necesario crear un objeto de negocio para poder gestionar la conexión del Web Service. El objeto de negocio creado es exactamente igual que el creado en el módulo básico de pedidos “kd_item”, pero en este caso se ha suprimido la consulta y se ha nombrado “kd_item_ws”. Al margen de este cambio, no hay ningún cambio más que quepa destacar.

Formulario: la creación del formulario consta de una parte sencilla, que ha sido copiar el formato del formulario de artículos y de otra que es la creación de una ventana modal[12] con los datos de configuración que se utilizaran para la conexión del Web Service.

Definidora de servicios web: Lo primero de todo para crear un Web Service es usar la definidora. Para ello solo hay que indicar el nombre del servicio, el objeto de negocio sobre el que actúa y la clase Java con la que se implementa el servicio.

A continuación se muestra una imagen de la definidora:

Figura 37: Definidora de Web Services

Clase de personalización: Los Web Services se configuran principalmente desde Java y es precisamente para esta historia para la que se ha requerido la versión de Eclipse con soporte para servicios web, tal como se ha comentado en el capítulo de entornos de desarrollo.

La primera clase que hay que implementar es la clase “KDWSItem”, que contiene las funciones necesarias para exportar los datos del objeto de negocio al archivo del servidor y las funciones que obtengan esos mismos datos del servidor para cargarlos en el formulario. Acto seguido se requieren una serie de clases tanto para la configuración del servidor como para el cliente. Todas ellas se encuentran en el CD del proyecto.

A continuación se muestra el formulario resultante de consulta de artículos por Web Service:

[12] Ventana modal: En términos de interfaces de usuario, una ventana modal es una ventana hija que requiere que el usuario interaccione con ella para poder volver a operar con la ventana madre.

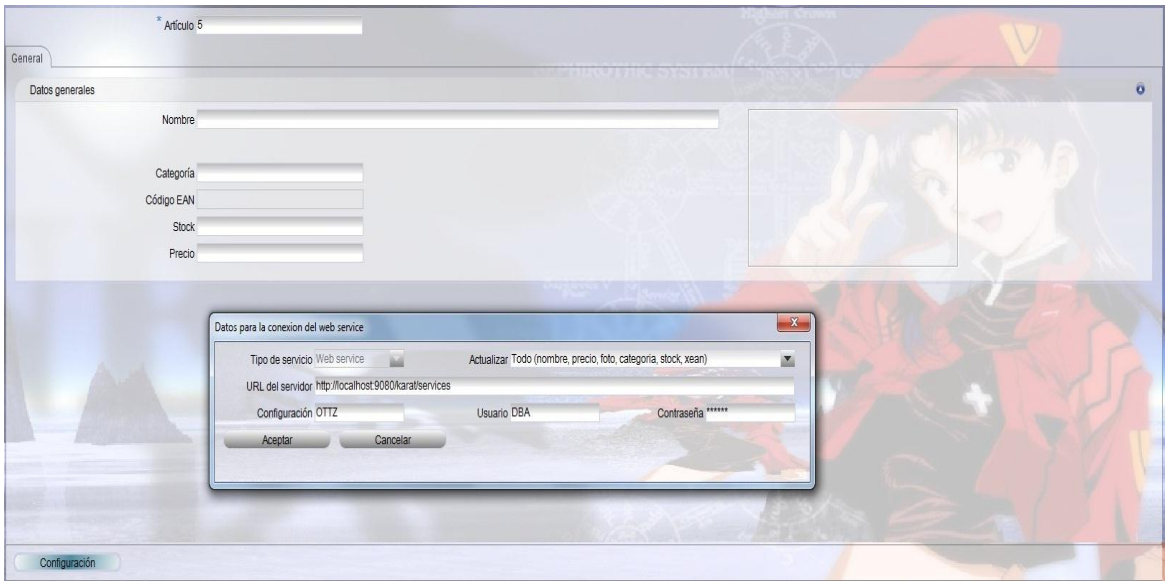


Figura 38: Formulario que vía Web Service muestra los datos de un artículo

- **Añadir cuadro de mando, uso de servicio, Widgets y Workspaces**

  La construcción de un Widget está formada por varios pasos, definir un servicio, usar el servicio para definir el Widget, agrupar Widgets para formar un Workspace e integrar el Workspace en la pestaña de cuadro de mando de la barra de herramientas de karat. A continuación se explica paso a paso la creación de un Widget en forma de velocímetro, que muestre el importe de las ventas del mes actual.

  Definición del servicio: La base para todo Widget es definir un servicio, que no es más que la función básica que se lleva a cabo, para su posterior adaptación a una interfaz visual más atractiva, la cual lleva a cabo a través de la definidora de Widgets. Así pues mediante la definidora de servicios se da nombre al servicio, se define sobre qué objeto de negocio se trabaja y si se desea que al hacer *clic* en el Widget se abra un formulario con los datos, también se debe especificar qué formulario.

  Una vez definidos esos valores solo queda especificar qué función de las posibles se desea ejecutar y sobre qué control, en este caso, suma del control que contiene el importe total del pedido.

  A continuación se muestra la definidora de servicios:

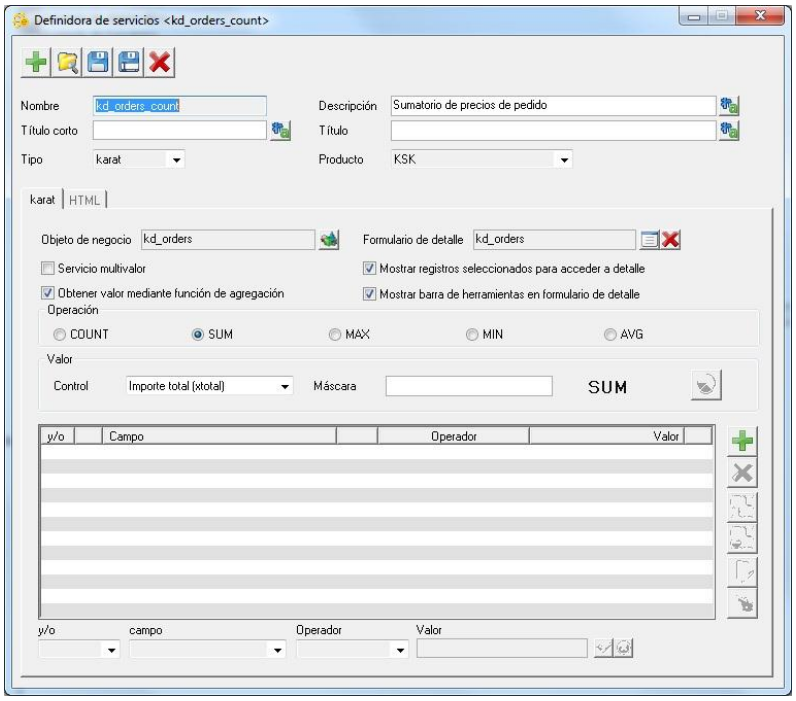


Figura 39: Definidora de servicios

Definidora de Widgets: Una vez definido el servicio, se abre la definidora de Widgets y se selecciona el servicio creado. Se añade un título corto al Widget y se especifica alguna restricción extra si se requiere, en este caso, que el control fecha sea igual al mes actual. Una vez definidas las restricciones se pueden seleccionar umbrales para cambiar el fondo del velocímetro a diferentes "velocidades".

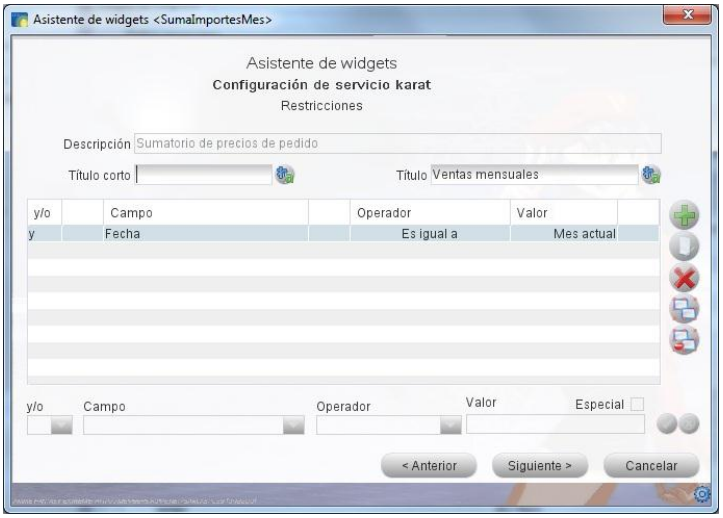


Figura 40: Definidora de Widgets

Creación del Workspace: A parte del Widget definido paso a paso, se han creado otros cuatro Widgets más, más sencillos, que constan sólo de un campo de texto y un campo numérico que indica la cantidad de objetos que cumplen la condición indicada (artículos con stock inferior a 10 unidades, ventas superiores a 10000€, cantidad de pedidos en estado "pendiente" y precio medio de todas las ventas realizadas por la empresa). Una

vez definidos todos los Widgets, solo hay que abrir el gestor de Workspaces y crear uno nuevo con todos los Widgets que se desea agrupar en él.

A continuación se muestra el escritorio con el Workspace creado:

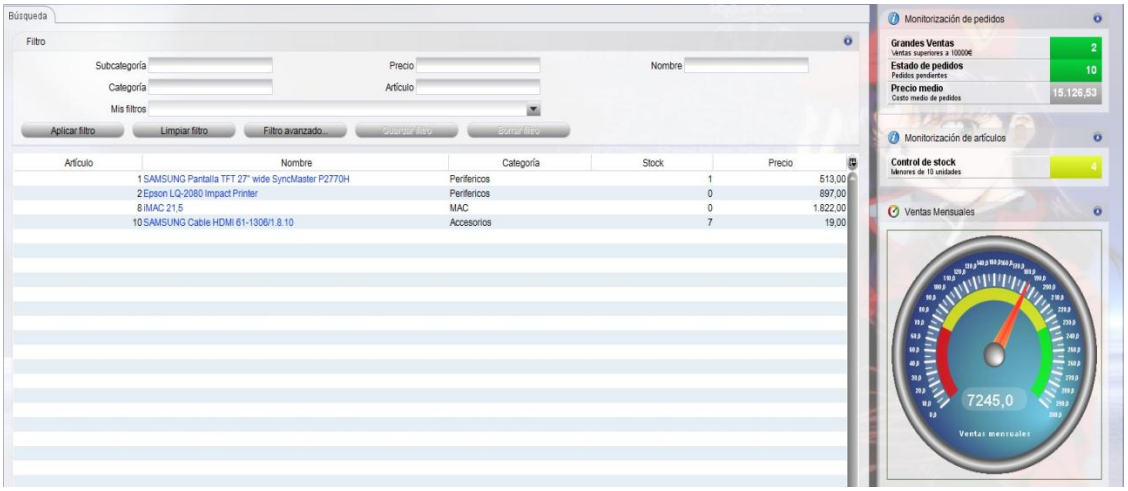


Figura 41: Workspace con todos los widgets

- **Ejemplo sencillo de karat Office Bridge**

Como se comentó en el Sprint 0 el formulario de pedidos contaba con una pestaña especial para karat Office Bridge. En esta pestaña se muestran las opciones con las que se pueden crear los documentos ofimáticos, desde una hoja de cálculo importando los valores directamente del formulario y aplicando fórmulas para calcular el importe con IVA del pedido, generar una hoja de cálculo usando una plantilla predefinida o generar un fichero de texto con una plantilla también predefinida. Para los documentos con plantilla, se ha habilitado un campo dónde se especifica el nombre de las plantillas que se generan automáticamente con la instalación del KSK.

A continuación se muestra la pestaña de karat Office Bridge:

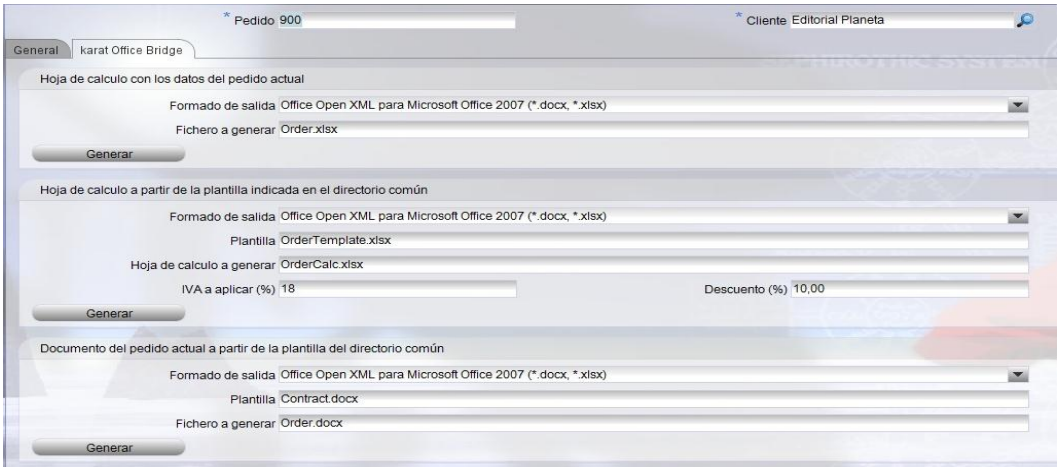
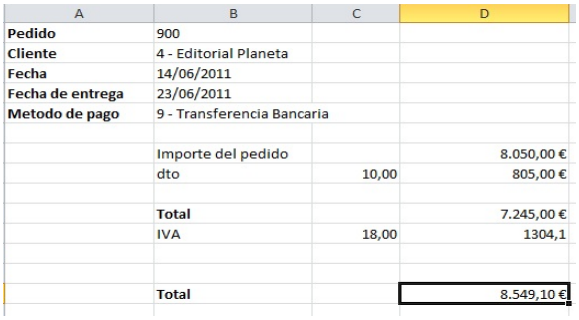


Figura 42: formulario karat Office Bridge

Clase de personalización: La clase de personalización es la misma que la del formulario de pedidos. De cara a karat Office Bridge hay dos grandes funciones: la primera se

encarga de autocompletar los campos referentes a los nombres finales de archivo, las plantillas y campo IVA, así como de gestionar qué tipo de fichero se selecciona en función de si se escoge un documento de Microsoft o de Open Office. La segunda función es la que trata directamente los datos en función de qué botón se ha pulsado, para generar el fichero final correspondiente y abrirlo inmediatamente después de su generación.

A continuación se muestra el resultado de generar una hoja de cálculo con una plantilla predefinida:



A	B	C	D
Pedido	900		
Cliente	4 - Editorial Planeta		
Fecha	14/06/2011		
Fecha de entrega	23/06/2011		
Metodo de pago	9 - Transferencia Bancaria		
	Importe del pedido		8.050,00 €
	dto	10,00	805,00 €
	Total		7.245,00 €
	IVA	18,00	1304,1
	Total		8.549,10 €

Figura 43: Hoja de cálculo generada con karat Office Bridge

- **Incorporar búsqueda rápida**

  En esta historia se pretende incorporar todos los datos de los formularios creados hasta ahora en la búsqueda rápida. La búsqueda rápida es una herramienta que ofrece karat para buscar, sin necesidad de ir a ningún formulario concreto, un dato que se desee desde la ventana principal.

  Incorporar la búsqueda rápida es tan sencillo como navegar por la definidora de objetos de negocio y seleccionar control a control cual queremos que aparezca en ella, de esta manera se incorporan en la memoria caché[13] todos los posibles contenidos de cada control seleccionado, de manera que al escribir una palabra se buscan automáticamente las coincidencias, aun sin haber terminado de escribir la palabra completamente.

  A continuación se muestra la búsqueda rápida:

Figura 44: Búsqueda rápida

- **Despliegue del nuevo producto y realizar copia de seguridad de las clases de personalización**

[13] Memoria caché: La caché es una memoria más pequeña y rápida, la cual almacena copias de los datos ubicados en la memoria principal utilizados con más frecuencia.

Para poder instalar la aplicación en todos los equipos que usan karat, se ha decidido crear un instalable para su fácil distribución y, a la vez, crear una copia de seguridad del código fuente de las clases de personalización para prevenir posibles pérdidas por daños en el equipo.

La creación de esta distribución y de la copia de seguridad ha sido transparente para mí, ya que la ha realizado mi tutor en la empresa con aplicaciones de las que yo no dispongo, lo único que he tenido que realizar es una copia de todos los archivos a una unidad de red para su posterior procesamiento.

5.4.3. Fase de prueba

La fase de prueba para este Sprint ha consistido en verificar que los datos cargados a través del Web Service fueran los correctos y que la información mostrada por los Widgets sea la correcta, además de que al hacer clic en ellos se redirigiera correctamente al formulario en cuestión.

5.4.4. Documentación

Como resultado de la fase de prueba se han generado dos ficheros, que se encuentran en el CD del proyecto, con los nombres: H0264_WebServices_VU y H0266_Widgets_VU.

5.4.4. Burndown chart

A continuación se muestra el Burndown-chart de este sprint y se comenta el resultado.

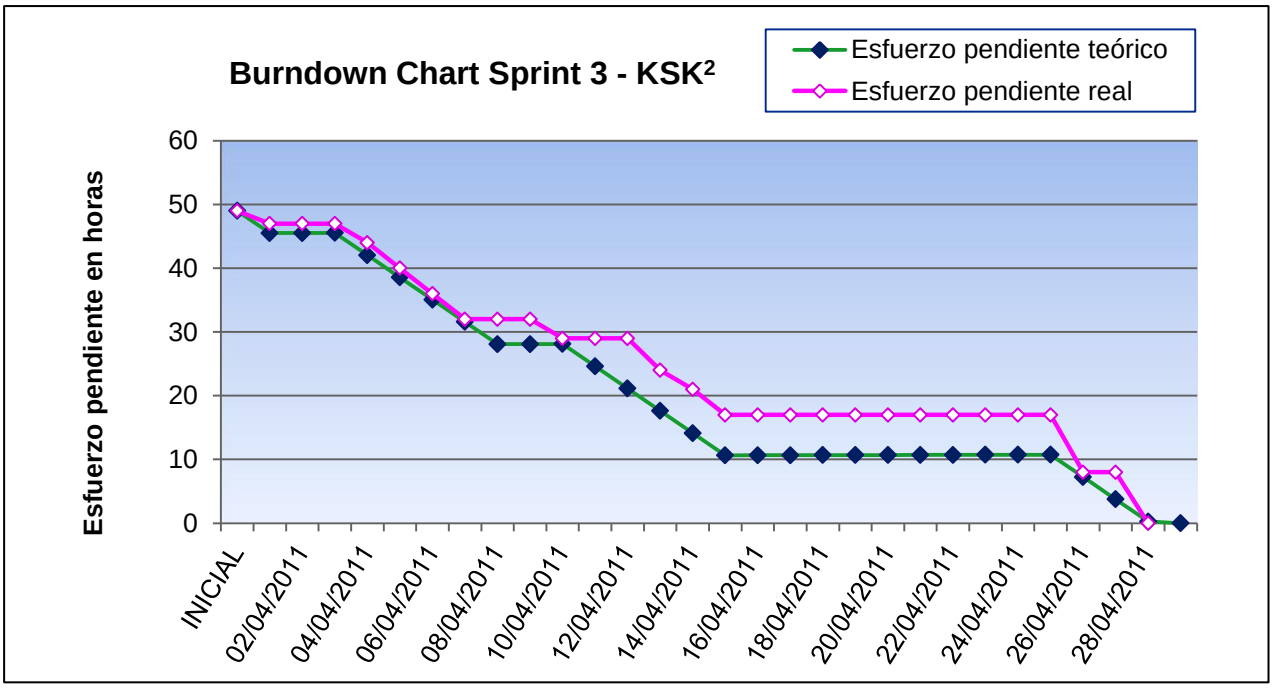


Figura 45: Burndown chart del Sprint 3

Como se puede observar en el Burndown chart al principio del Sprint las rectas han transcurrido muy similares, con la única desviación a partir del día 11 ya que por motivos personales me he ausentado del trabajo, lo cual se refleja en la línea recta a medio gráfico y la línea más larga corresponde a semana santa. Al final he dedicado un esfuerzo extra para poder cumplir con todos los objetivos marcados por este Sprint.

5.5. Sprint 4

5.5.1. Tareas a realizar

Historia	Descripción	Coste
H0273	Documentación de usuario del nuevo producto.	2,00
H0275	BUGS Sprint 4	0
	Total	2,00

Tabla 14: Historias del Sprint 4

Como se puede observar en la tabla de historias, este Sprint contiene únicamente un gran bloque que corresponde a la documentación y ayuda integrada del KSK.

5.5.2. Fase de codificación

Para la generación de la ayuda online se ha hecho uso del proyecto en fase Beta de un compañero que se ha dedicado precisamente a la mejora de la generación de ayudas.

Para generar una ayuda se ha usado un manual interno: "¿Cómo generar un Cómo…?". En él se recogen las instrucciones a seguir, en cuanto a estructura, para generar una ayuda de tipo *Cómo* que siga la normativa de la empresa.

A través de la herramienta utilizada, "Unit4Help", se han generado los ficheros HTML que forman la ayuda. Cómo toda ayuda para una aplicación, ésta sigue una estructura de árbol con un nodo raíz y a partir de ahí cuelgan otros archivos con el contenido de cada archivo del que depende.

Se ha generado un fichero HTML por cada formulario que forma el KSK donde se explican los conceptos básicos del contenido del formulario y si en éste se encuentra alguna funcionalidad especial, un enlace al documento que contiene dicha ayuda.

Para el caso del Simple Workflow se ha generado una documentación específica de cómo generar una regla de negocio que verifique los pedidos con importe superior a 6000€.

Una vez generados los archivos de ayuda, se tienen que enlazar a los formularios de manera que cuando pulsemos la tecla F1 desde uno, se abra el documento que contiene la ayuda específica de ese formulario. Esto se consigue mediante los *identificadores de tópico*. Un identificador de tópico no es más que una referencia única (en el caso del KSK una referencia numérica) que se asigna a un archivo HTML, de manera que cuando se quiera acceder a ese documento habrá que especificar ese identificador. Una vez asignados los identificadores a todos los archivos HTML ha sido necesario a todos los formularios del KSK

y, a través de las propiedades del formulario, asignarle el identificador de tópico, de esta manera se enlaza totalmente un formulario con un archivo de ayuda.

El último paso para que la ayuda funcione correctamente es copiar los archivos generados por la aplicación en una carpeta del ordenador donde se encuentran todas las ayudas de todos los módulos. Como cabe pensar, si hay varias ayudas es posible que karat no sepa a cuál de ellas tiene que dirigirse ya que puede haber varias con el mismo identificador de tópico. Esto se soluciona a través de una pequeña función que se añade a las clases de personalización de los formularios que ya disponían de una, y creando una clase de personalización para los formularios que no disponían porque no había sido necesario hasta ahora. Esta función lo único que hace es decir a karat en que carpeta se encuentra el identificador de tópico al que hacemos referencia al pulsar F1. De esta manera se evita que acceda a carpetas con documentación de otros módulos.

A continuación se muestra un ejemplo de cómo se ve la ayuda tras pulsar F1 en el mantenimiento de pedidos.

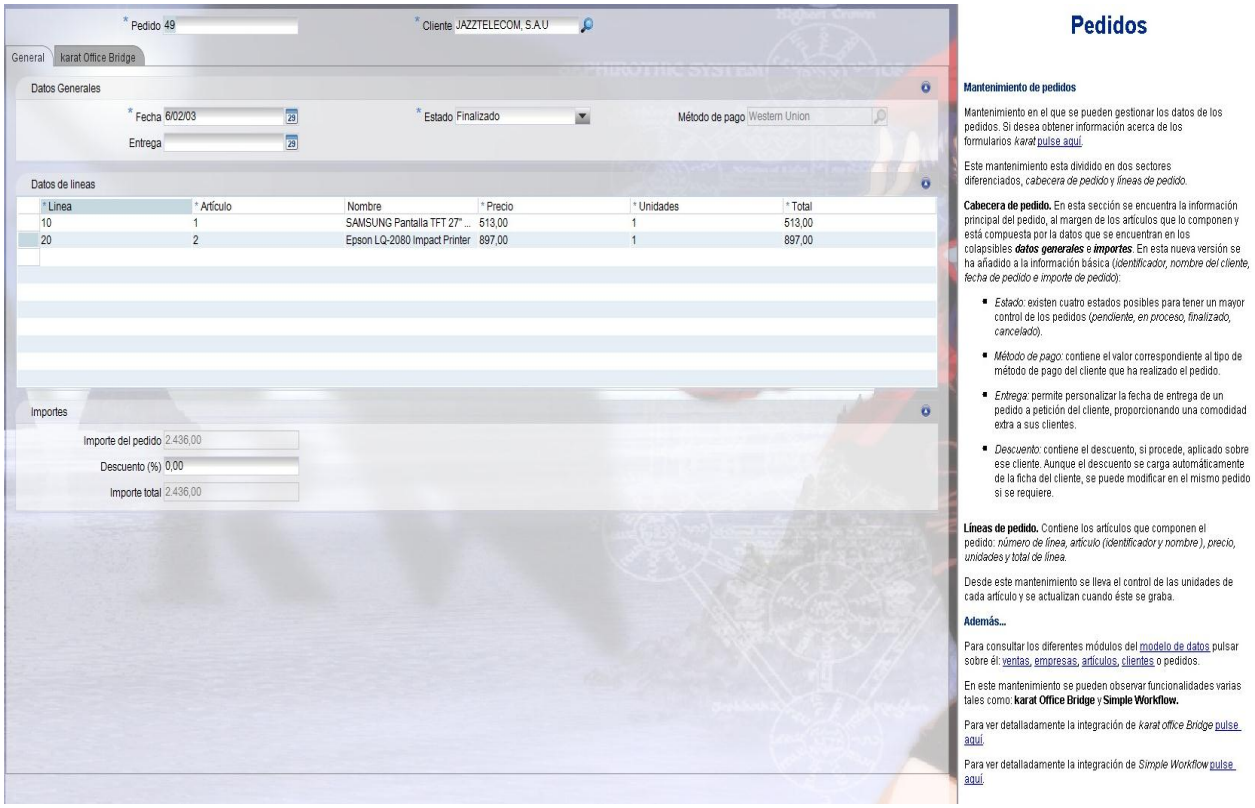


Figura 46: Mantenimiento de artículos tras pulsar F1

5.5.3. Fase de prueba

La fase de prueba para este Sprint ha consistido en comprobar que los enlaces funcionen correctamente y que se muestren las ayudas correspondientes a los formularios desde los que se solicita la ayuda.

5.5.4. Documentación

En el CD del proyecto se adjunta toda la documentación generada durante este Sprint, que corresponde a todos los archivos HTML que la componen. La documentación creada para este Sprint se adjunta en el CD del proyecto

5.5.5 Burndown Chart

Para este Sprint no hay Burndown chart debido a que no sólo se ha dependido del trabajo realizado generando documentación sino que al usar una herramienta en fase Beta ha sido necesario en algunos momentos esperar a la resolución de los problemas encontrados. Además este Sprint también ha coincidido con la fase de exámenes finales con lo que a causa de estos dos sucesos la curva de trabajo se ha visto muy alterada y a mediados del periodo que comprende este Sprint se decidió no seguir con el gráfico ya que no mostraba un resultado útil cara al estudio del progreso.

CONCLUSIONES

6.1. Objetivos alcanzados

Una vez finalizado el proyecto llega el momento de observar si los objetivos alcanzados corresponden con los que se estimaba en su inicio.

Es evidente, con la lista de funcionalidades a mostrar en mano, que no se han podido incorporar todas, pero llegados a este punto y partiendo del conocimiento adquirido durante estos meses de trabajo en el desarrollo del proyecto, cabe decir que era de esperar que no se alcanzara tal envergadura. Los motivos que me llevan a afirmar esto, no son más que el fruto de la experiencia adquirida durante estos meses en UNIT4.

Como se ha podido observar en las tablas de las historias a realizar, en cada Sprint hay una historia de entre 0.25 y 0.50 puntos de formación acerca de cada funcionalidad a mostrar. Así pués me atrevo a decir que la propia naturaleza de mi proyecto, que engloba una gran variedad de funcionalidades y conlleva un continuo aprendizaje de la herramienta, implica que sea dificultoso para alguien que ha partido de un conocimiento muy básico de karat, como es mi caso, poder incorporar todas las funcionalidades previstas inicialmente.

Dejando de banda ese detalle, se puede afirmar que el producto resultante satisface los requisitos iniciales ya que se ha creado un modelo de ventas sólido desde sus raíces, siguiendo estrictamente los criterios Walnut y Java para karat, así como el soporte multiidioma y la documentación de usuario.

A continuación se muestra una lista con las funcionalidades que se quería mostrar inicialmente y cuales se han podido implementar finalmente:

- ✓ Módulo básico de pedidos.
- ✓ Listados e impresos con plantilla
- ✓ Modelo Organizacional con unos roles definidos
- ✓ Simple Workflow
- ✓ Dashboard
- ✓ Collaboration & Office
- ✓ Web Services, XML
- ✓ Documentación
- \+ Creación de escritorios
- \+ Búsqueda rápida
- \+ Despliegue del nuevo producto
- × Fórmulas
- × Remote Call
- × Informes navegables
- × Transferencia de datos

Las funcionalidades con el símbolo “+”, son funcionalidades que se han introducido a medio proyecto y que inicialmente no estaban planificadas.

6.2. Propuestas de ampliación.

Una vez vistos los objetivos alcanzados, se puede deducir fácilmente que una propuesta de ampliación pasa por acabar de integrar las funcionalidades que no se han integrado hasta ahora (fórmulas, remote call, informes navegables y transferencia de datos) y las que, al igual que ha sucedido con los escritorios que no constaban en la definición inicial, puedan ir surgiendo fruto del crecimiento de la plataforma karat.

Al margen de la adición de las funcionalidades que faltan, sería necesario crear una versión en inglés de la documentación, ya que en contraposición a los pequeños textos que se han introducido en los objetos de negocio, la documentación está formada por textos extensos y requieren de documentalistas especializados en generación de documentos en este idioma.

Quitando funcionalidades importantes también sería importante refinar los mantenimientos existentes con pequeñas mejoras que no añaden valor, pero que proporcionarían una interfaz algo más sencilla, ya que durante la fase de codificación el enfoque ha sido más encarado a añadir funcionalidades que a refinar en detalle cada mantenimiento.

6.3. Desviaciones de la planificación

En este sentido poco hay que decir de la planificación del proyecto que no se haya explicado ya en los correspondientes Sprints.

Es evidente que ha habido una pequeña desviación en cuanto a la planificación inicial, pero como se ha comentado en los apartados de Burndown chart, ésta ha sido debida ya sea a que la historia de los Web Services resultó más complicada de lo que se esperaba inicialmente, como al hecho de compaginar el proyecto con las últimas asignaturas de la carrera, lo cual también se ha reflejado cara a la preparación de algunos exámenes.

No obstante, como ya se ha comentado, la desviación ha sido pequeña y bajo mi punto de vista esperable y sin demasiada repercusión en un proyecto de esta envergadura.

6.4. Valoración personal

La realización de este proyecto ha supuesto un reto personal en todos los sentidos.

Hasta ahora todas las prácticas que había realizado habían sido de una envergadura menor y eran fruto del trabajo en equipo de dos o tres personas, sin contar con que estaban fuertemente guiadas por los profesores de la universidad. Con la realización de este proyecto me he visto por primera vez con un proyecto de una gran complejidad delante de mí y teniendo en cuenta que, aunque me han guiado durante el transcurrir del proyecto, era responsabilidad mía planificar la gestión del proyecto en todas sus fases.

Al margen de este aspecto, ésta ha sido la primera toma de contacto con el mundo laboral en el sector de la informática y me ha servido, en un grado muy satisfactorio, para ver cómo es la atmósfera de trabajo en una empresa de diseño de software cuyo trabajo está muy marcado por el estricto cumplimiento de unas fechas de entrega y que, hasta ahora, solo había podido imaginar mientras cursaba mis estudios.

Gracias a este proyecto y a la oportunidad que me han brindado tanto la Universitat Autònoma de Barcelona como la empresa UNIT4 he perdido ese miedo que todos tenemos la primera vez que entramos en un mundo laboral nuevo, ya que aunque el primer día veía la finalización del proyecto como una montaña muy pronunciada y difícil de escalar, ahora me doy cuenta que con trabajo diario y con la formación que he recibido durante estos años, acompañado por supuesto del asesoramiento de expertos en la materia, todo proyecto es realizable.

Así pues me complace decir que estoy totalmente orgulloso de haber tomado la decisión de intentar acceder a estas plazas que la UAB y UNIT4 ofrecen a los estudiantes en búsqueda de un proyecto final de carrera, porque a pesar de que las horas dedicadas a éste son ampliamente superiores a las que se dedicaría a un proyecto por cuenta propia, la recompensa personal, profesional y sobretodo moral que supone, no tiene precio alguno.

BIBLIOGRAFIA

A continuación se muestran las referencias bibliográficas que se han usado para realizar el proyecto, aparentemente pueden parecer pocas pero dada la naturaleza del proyecto lo que más se ha consultado es documentación interna de la empresa, que se ha completado con búsquedas puntuales por internet para algún aspecto más general.

- Wikipedia, enciclopedia libre y políglota de la Fundación Wikimedia [en línea] [último acceso Junio de 2011]. Disponible en Web: http://es.wikipedia.org

- Blue Claw Database Design, página web con ejemplos de bases de datos de varios tipos y temáticas [en línea]. Idioma: Inglés [último acceso Junio de 2011]. Disponible en Web: http://www.blueclaw-db.com/screen_samples/

- Tutoriales sobre las funcionalidades karat en formato pdf ubicados en la intranet de la empresa.

- Documentación oficial de karat con API de referencia incluida ubicada en la intranet de la empresa.

AGRADECIMIENTOS

En este apartado me gustaría agradecer a todas las personas que me han ayudado a llevar el proyecto a buen puerto y sin cuya ayuda no habría sido capaz de alcanzar los objetivos deseados.

- A Eduard Mestre como tutor en UNIT4 que me ha guiado en mi paso por la empresa y que a su vez ha realizado el rol de Scrum Master guiándome y asesorándome en las reuniones diarias.

- A Ezequiel Parra como Jefe de proyecto en la empresa, por su visión crítica durante las demostraciones realizadas y su orientación cara a la defensa del trabajo realizado.

- A Jordi Pons como tutor por parte de la universidad por su enfoque más académico del proyecto y por revisar de principio a fin la memoria del proyecto.

- Al equipo de platform de UNIT4 en concreto al departamento de *back* por asesorarme en momentos puntuales durante el proyecto.

Carlos Ming Martínez

a 22 de Junio de 2011